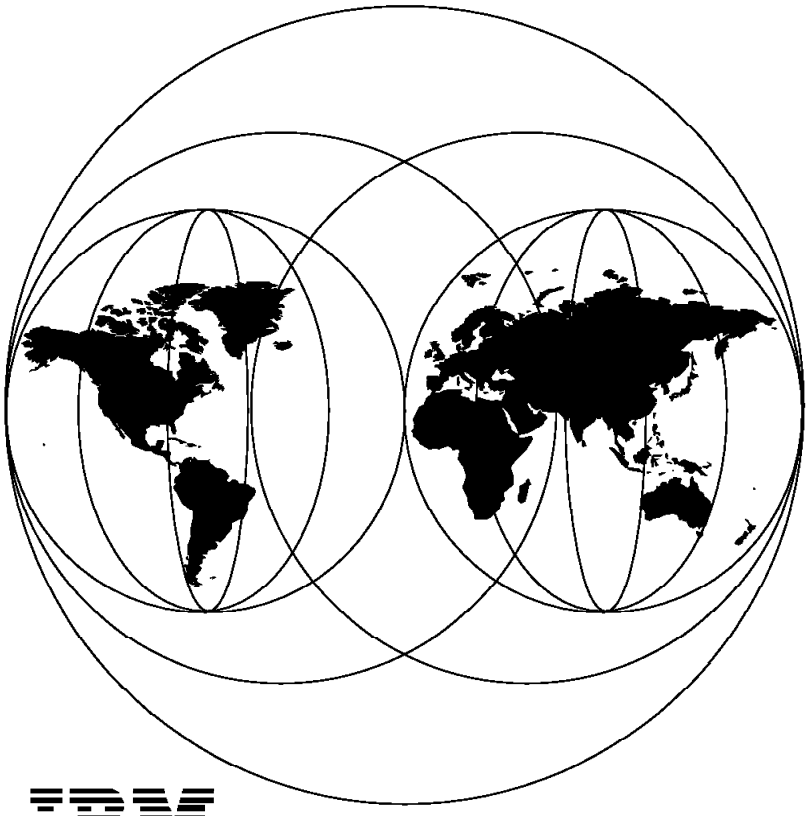
SG24-4848-01

The Domino Defense:
Security in Lotus Notes and the Internet

December 1997



IBM

International Technical Support Organization
Raleigh Center

International Technical Support Organization

SG24-4848-01

The Domino Defense: Security in Lotus Notes and the Internet

December 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 155.

Second Edition (December 1997)

This edition applies to Version 4 Release 5 of Lotus Notes.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1997. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1997

Figures

© Copyright IBM Corp. 1997

Tables

© Copyright IBM Corp. 1997

Preface

The Domino server joins together the secure and well-managed environment of Lotus Notes with the universal access of the World Wide Web. This redbook helps the reader understand the security facilities provided by Lotus Notes 4.5 and by the World Wide Web and then illustrates the value that Domino Web Server 1.5a brings to the area of security by joining together these two environments. The redbook provides an example application to show a practical implementation of these concepts. The book also assists the reader in determining the correct placement of Domino servers within an Internet firewall, again using detailed examples.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Rob Macgregor is a systems engineer at the Systems Management and Networking ITSO Center, Raleigh. He writes extensively and teaches IBM classes worldwide on the subjects of Internet security and open systems management. Before joining the ITSO three years ago, Rob worked in the UK as a technical support specialist.

Carla Sadtler is a Senior ITSO Specialist for Communications Server for AIX and network integration projects at the Systems Management and Networking ITSO Center, Raleigh. She also has responsibility for Lotus Notes and MVS system programming in the ITSO Center in Raleigh. Before joining the ITSO 12 years ago, Carla worked in the Raleigh branch office as a program support representative.

Wang Xian Zhe is an I/T Availability Professional Specialist in IBM China. She has been supporting various kinds of software for two years, and also does platform system integration. Currently, her main focus is on TCP/IP, VisualGen and Lotus Notes.

Hidenobu Yamamoto is an I/T specialist at IBM Japan Systems Laboratory. He joined IBM in 1993 and has been working in various areas of technical support since then, recently focussing on Lotus Notes and Internet integration and security, including PC, UNIX, and mainframe environment. He participated in the writing of several redbooks for the ITSO Centers in Dallas and Raleigh.

Thanks to the following people for their invaluable contributions to this project:

David Boone, Paul Braun, Shawn Walsh, and Linda Robinson
Systems Management and Networking ITSO Center, Raleigh

Takeo Shibazaki
IBM Japan

Charlie Kaufman
Iris Associates

Alan Eldridge
Iris Associates

© Copyright IBM Corp. 1997

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 165 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

For Internet users	http://www.redbooks.ibm.com
For IBM Intranet users	http://w3.itso.ibm.com/redbooks

- Send us a note at the following address:

  redbook@vnet.ibm.com

Chapter 1. Introduction

Unless you have been asleep for the past few years, you will have noticed that the world is becoming increasingly *network-centric*. It is possible that this phenomenon is a reflection of marketing hyperbole, rather than reality. However, we prefer to believe that this time, just possibly, it is true. There are several reasons for this optimism:

Luck In some cases, systems have been rescued from their problems by new technology. For example, many ambitious distributed application projects have foundered because the system resources they required were unattainable or uneconomic. Recent microprocessor and memory developments have been combined to deliver hardware that can handle these extravagant loads at a reasonable cost.

Evolution Platforms and development environments for distributed applications have tended either to not be very scalable (typical LAN-based network operating systems for example) or to be rather too complex for most customers (DCE comes to mind). The result has been that they have not delivered on their promise. However such environments have been gradually evolving to address these problems. Lotus Notes is a good example of this; it started as an ingenious groupware platform that was very effective in a small environment. Over the past years it has developed out of that role, gaining much improved scalability and flexibility along the way.

Revolution Without a doubt the World Wide Web has had a revolutionary effect on the distributed applications scene. It has not only wakened the whole world to the possibilities that a public access global network offers, but has also given developers new, simpler, facilities for developing distributed applications.

One of the first questions that is asked about the Web, once the initial excitement has worn off, is “Is it safe?” There is no simple answer to this, because there is no absolute measurement of what constitutes “safety.” The best response is to understand the risks and the tools available to counter them, and then use that knowledge to reduce the risks to an acceptable level.

In this redbook we look at security aspects of two network-centric environments, Lotus Notes and the World Wide Web, and then consider what happens when the two of them are molded together in the form of the Lotus Domino server. Before we look into the details, let us consider the reasons why we are interested in security, that is what are we trying to achieve and what risks are we trying to address?

1.1 Security Objectives

When you consider the design of any computer system there are always security requirements to be addressed. These requirements differ wildly from one system to another, but they can all be broken down into a number of categories, or services, that the system must provide. The categories that we use in this redbook are taken from the IBM Security Architecture, which is based on ISO 7498 security standards.

© Copyright IBM Corp. 1997

The IBM Security Architecture is a model for integrating security services, mechanisms, objects, and management functions, across multiple hardware and software platforms and networks. The architecture supports the strategy for providing end-to-end protection of applications and information within an organization. In this redbook we focus on the basic security services and mechanisms provided by the Notes and Domino systems, but you should keep in mind that any system, no matter how well-conceived, is only as good as its implementation and administration allows. The security services and mechanisms are only part of the complete story. We divide the services into the following categories:

- Access control
- Identification and authentication
- Confidentiality
- Data integrity
- Non-repudiation

The following descriptions of these security services are drawn from *Enterprise-Wide Security Architecture and Solutions*, SG24-4579. Keep in mind that the categories are not exclusive; for example, you cannot do access control without also addressing questions of authentication and data integrity.

1.1.1 Access Control

Access control allows the installation to protect critical resources by limiting access to only authorized and authenticated users. Depending on the environment, access may be controlled by the resource owner, or, it may be done automatically by the system through security labels. The resource owner can specify who can access the information, how it can be accessed, when it can be accessed, and under what conditions it can be accessed (for example, when executing specific applications, programs, or transactions). The functional goal is to assure that security is maintained for resources, whether they are in a central system, distributed, or mobile (as in the case with files and programs).

1.1.2 Identification and Authentication

Identification and Authentication (I&A) facilities verify the identity of individuals. The basic function uniquely identifies users and programs, verifies these identities, and assures individual accountability. Authentication may be single authentication, for an individual user to the system, mutual authentication of peers, such as two party authentication for distributed applications, or three party authentication when dealing with local authentication servers in a distributed environment.

Authenticated user identification provides the basis for additional security functions, such as access control and auditing. Authentication technology may take the form of passwords, smart tokens, smart cards, and biometric measuring devices.

1.1.3 Confidentiality

Confidentiality protects sensitive information from disclosure. When it is stored locally, sensitive data can be protected by access controls or encryption mechanisms. For network communication security, sensitive data should be encrypted as it is transmitted from system to system. There are specific ISO standards (8730, 8731, and 9564) relating to use of cryptography for confidentiality and data integrity.

1.1.4 Data Integrity

Data integrity provides detection of the unauthorized modification of data. Organizations must allow for the use of data by authorized users and applications, as well as the transmission of data for remote processing. Data integrity facilities can indicate whether information has been altered. Data may be altered in two ways: because of hardware or transmission errors or because of an attack. For years, many IBM products have used a check sum mechanism in disk and tape storage systems and in network protocols to protect against transmission and hardware errors. Active attacks on data integrity require a different mechanism, which uses cryptography and allows for the verification of data integrity.

To address active attacks on data integrity, products must support message authentication based on cryptographic functions that adhere to international standards.

1.1.5 Non-Repudiation

Non-repudiation may be viewed as an extension to the identification and authentication services. The non-repudiation service can protect a recipient against the false denial by an originator that the data has been sent, and it can protect an originator against the false denial of a recipient that the data has been received. In general, non-repudiation applies to the transmission of electronic data, such as an order to a stock broker to buy/sell stock, a doctor′s order for medication to a specific patient, or approval to pay an invoice by a company to its bank. The overall goal is to be able to verify, with virtually 100% certainty, that a particular message can be associated with a particular individual, just as a handwritten signature on a bank check is tied back to the account owner.

Chapter 2. The Domino Server

The Lotus Domino server is a combination Notes server and Internet server, allowing Web browsers dynamic access to Notes applications. This offers to users the global access and rich media of an Internet environment combined with the wonderful data organization and application development features of Notes. This chapter takes a very brief look at how Domino works. More detailed information can be found in *Working with Lotus Notes and the Internet*, which is shipped with Domino. Information is also available at http://domino.lotus.com.

The Lotus Domino server is a Notes server with the same functions and components as previous Notes servers. In addition, Domino provides an HTTP server component to enable Web clients to communicate with the server and a URL interface with extensions that allow the Web client use of the Notes database components. Notes database components such as views, documents, links, navigators, and buttons are dynamically translated to HTML for URL requests.

Domino also has access to the kind of files that any normal Web server would use. For example you can imbed HTML within Notes documents to control document formatting and links. You can also make use of graphic and multimedia formats such as GIF, JPEG and any other format that your browser can handle. In addition to the programmability provided by Notes itself, the Domino server provides you with the normal programming interfaces for Web servers and browsers, such as CGI and Java.

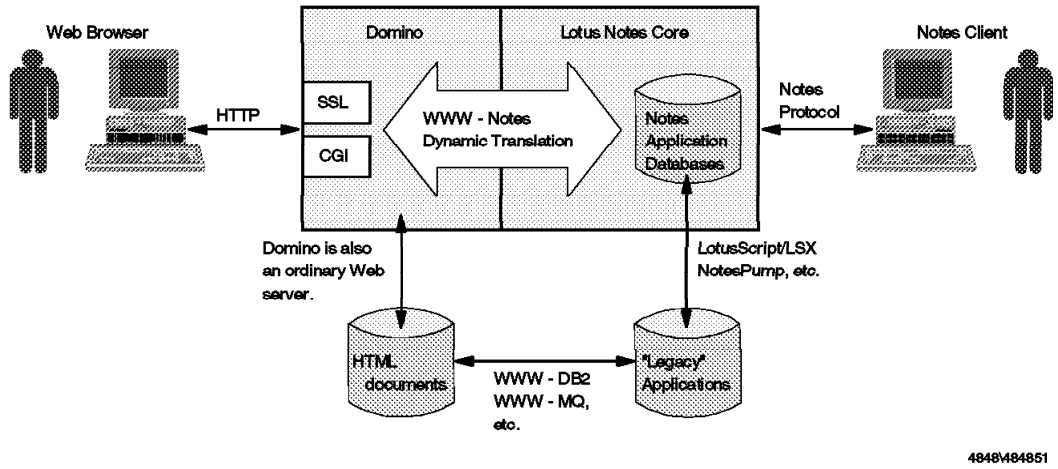


Figure 1. Domino Server

An experienced Notes administrator will find that supporting Domino is familiar because all of the functions are defined in the Notes Public Address Book server document. There are two changes to the server document for Domino. A new HTTP server section is added to define basic Domino information and the security section has new fields to determine the basic access of Web clients and to define Secure Sockets Layer (SSL).Figure 2 on page 6 shows an example of these server document sections.

© Copyright IBM Corp. 1997

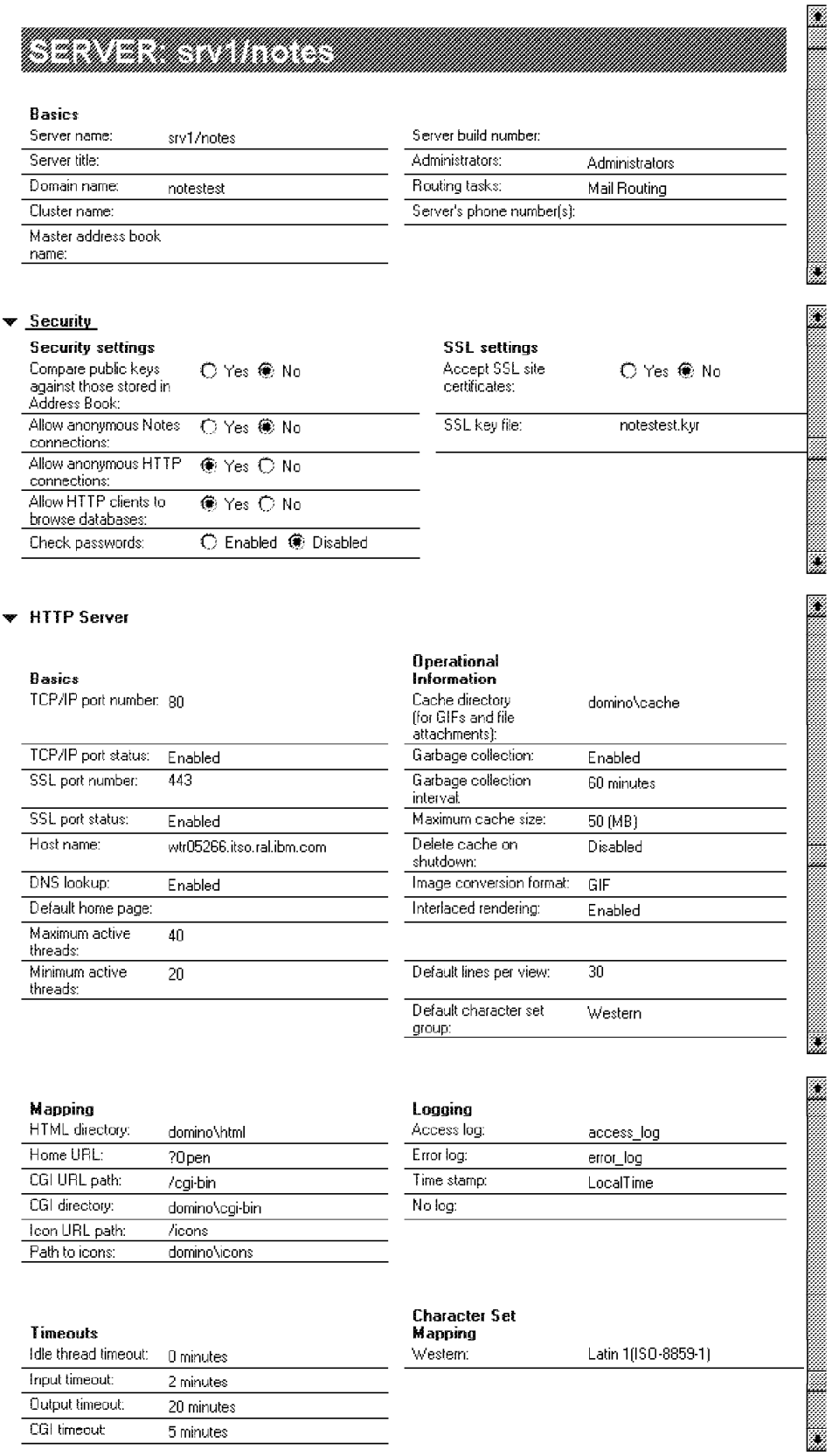


Figure 2. Domino Additions to the Server Document

2.1 A Typical Domino Site

Designing and maintaining a Domino server can be as simple or as complex as you wish. A typical Domino site will have a home page that Web users will see when they request the server. What the user sees is determined by the value of the Home URL field in the server document. The default value, ?Open, will present the user with a list of databases on the server. This field in the server document can be seen in Figure 2 on page 6. You can change this to present a navigator or view by putting the URL for that object in the Home URL field.

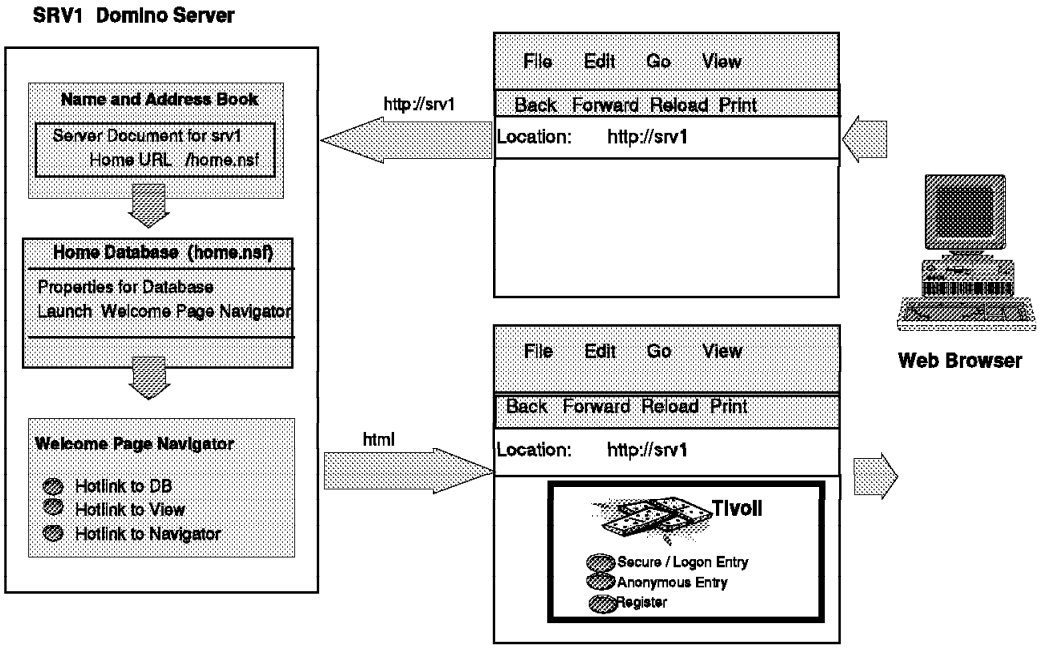


Figure 3. Domino Server Initial Web Page

In Figure 3 we show an example where the Home URL field has been updated with the URL of the home.nsf database. In turn, the properties for home.nsf have been set to launch a Welcome Page navigator. Now, imagine that a Web user enters the URL of the server without specifying a file name (for example, http://server.ip.address). The user will see the Welcome Page navigator, which includes a series of hypertext links to other URLs. Within Notes these links are really connections to other databases, views, or navigators. In fact, in the example the Welcome Page navigator has been designed in Notes using a background graphic with link hotspots . The link hotspots have been translated by Domino into HTML anchor tags, leading to other URLs.

2.2 How Domino Works

Domino translates Notes application components for delivery to the Web client. HTML combined with Domino URLs are created to represent Notes objects to a Web browser. HTML can be coded directly into formulas or Notes functions can be used and translated by Domino. Navigators are translated to image maps and graphics are translated to GIF or JPEG files.

When a Web client opens a Domino server by typing `http://servername`, the URL coded in the Notes server document Home URL field will be used to determine what to show the user. The default is the ?Open Domino URL. This will show a bulleted list of databases on the server. Each bullet is a hypertext link to a database.

When a user selects the URL for a database with no additional launch specifications, Domino will show a bulleted list of the views in the database. Each bullet is a hypertext link to a view. Often, a database will be defined to launch a navigator or the ″About Database″ document.

Forms, documents, and views are dynamically translated into HTML for display to the Web user. Formulas are calculated at the time the object is presented. Actions defined in Notes forms and views that are included in the Notes action bar are translated to HTML and included at the top of the Domino page.

Form processing is a good example of where the different capabilities of a Web browser and a Notes client may lead to a change in application design. An HTML form is very simple in construction, comprising of a number of fields that the user can type in or select with the mouse. The fields include various input gadgets (radio buttons, selection lists, text areas, etc.) but the operating paradigm is that the form contains only two action buttons, a submit button that sends the filled-in form to the server for processing and an optional reset button that clears all entries. Notes allows multiple buttons on a form, each invoking a different action. The way that Domino handles this discrepancy is to ignore all except for the first button on the form, which is assumed to be the submit button. Any formula for this button is ignored but the user will see the text. If you do not have a button on a form, a submit button is automatically generated when Domino translates the form to HTML. If a button is on the form, Domino associates the submit action with that button. (When the user clicks the button the document is submitted (saved).)

Attachments to Notes documents are supported. The graphic representing the attachment is translated to a GIF file. When the Web user clicks on the image a file attachment URL is created so the the document can be downloaded. If the attachment needs to launch a helper application in the browser the normal process applies, in which the browser maintains a mapping between file extension, MIME type and the action to perform.

CGI scripts, Perl scripts, Java, Java Script, VBScript are all supported to allow the database designer to build programming into Notes applications.

There are some extensions to the Notes design function that are specific to HTTP access. These are mainly special $$fields on forms. For example, the $$Return field on a form allows the designer to determine what actions or messages occur when a user submits a form.Figure 4 on page 9 and Figure 5 on page 9 show a simple example. When the user submits the form, the contents of the $$Return field cause the user to be presented with a message stating that the question has been submitted. The user can click on the text **Return to Question and Answer View** to return to that view.

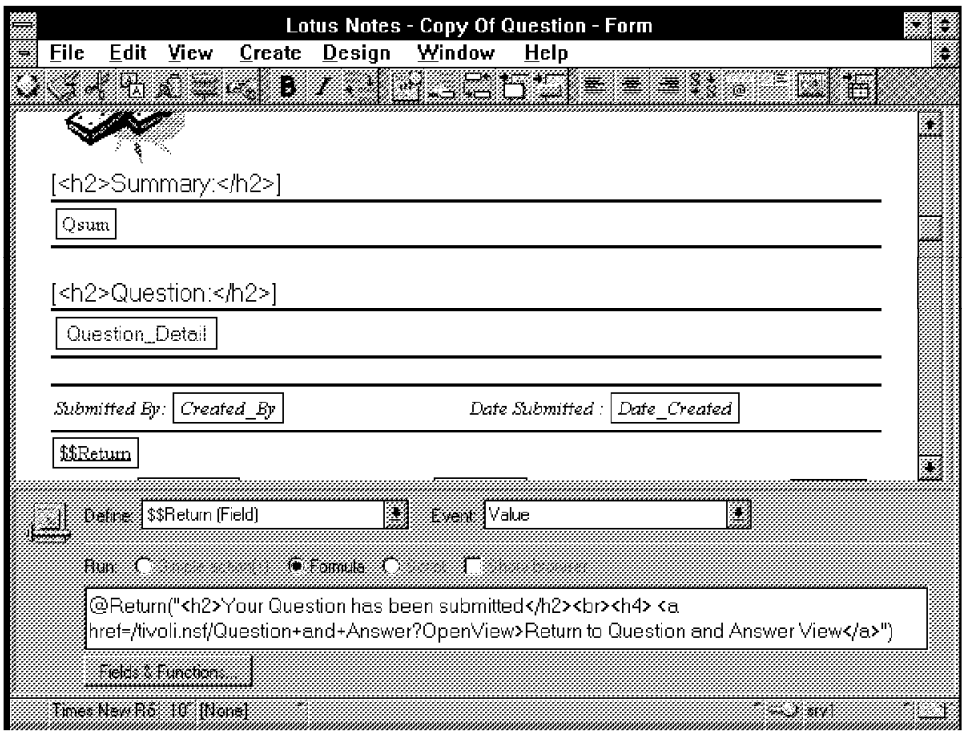


Figure 4. $$Return Field

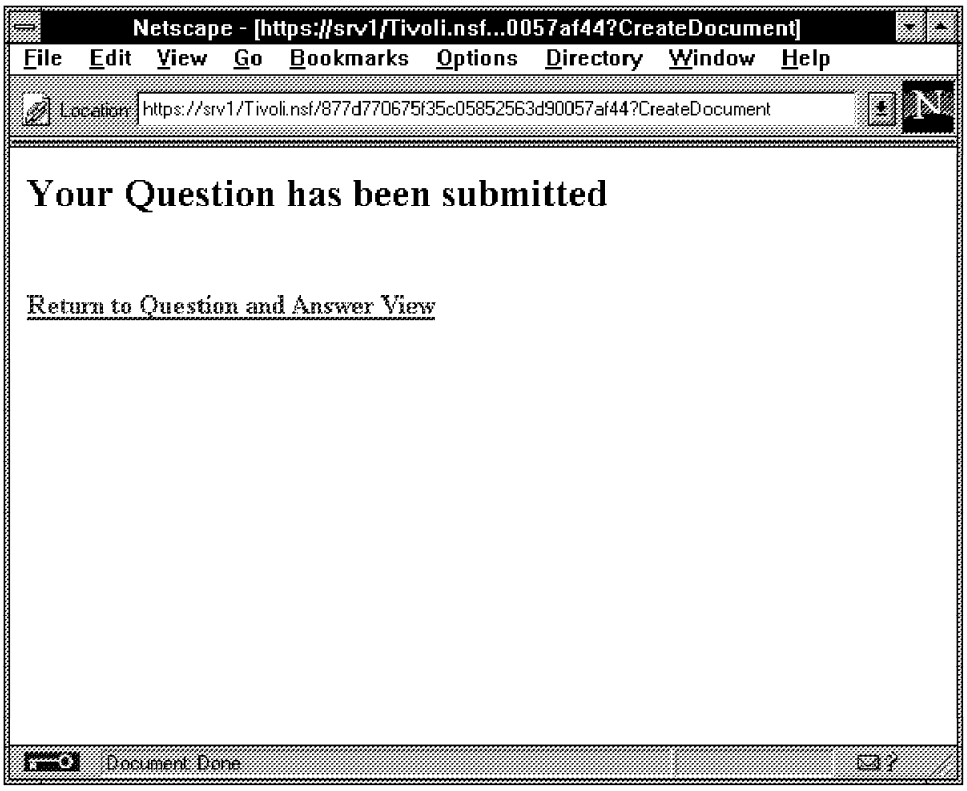


Figure 5. Message Presented after Submitting the Question Form

2.2.1 Domino URLs

As a Web browser user, you navigate among forms, documents, views, navigators and databases on a Domino server using Domino URLs, either by entering them directly or by clicking on a link. Domino URLs have the form:

```
http://server/database/NotesObject?Action&Arguments
```

Links of this kind to Notes objects can be coded into Notes applications in many ways. They can be invoked by Notes commands, simple actions available when defining hotspots, actions, and agents, or by coding HTML statements directly into Notes forms. Domino then takes the links and translates them into Domino URLs.

When Domino builds a URL to link to documents and views it constructs a hierarchy of references using the universal ID (also called UnID) of the document or view. For example, if you click on a database record in a view, the URL that Domino directs you to will have the following format: http://server/db.nsf/view UnID/document UnID?OpenDocument.

Figure 6 illustrates the flow of URLs created when a Web client accesses a Notes database on a Domino Server.

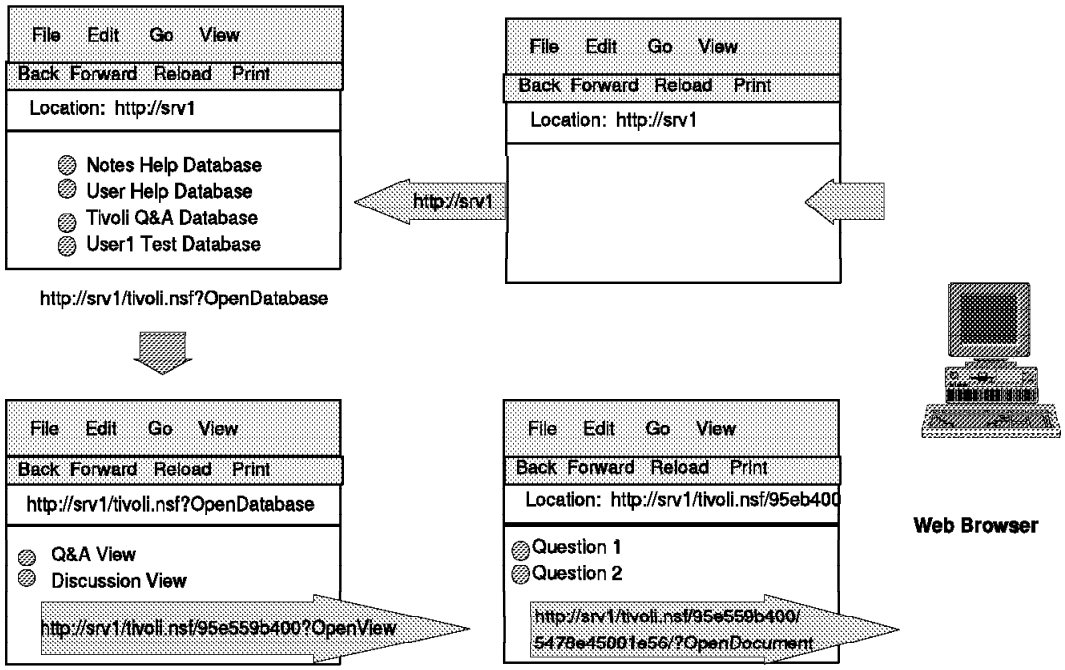


Figure 6. Domino URLs

In Figure 6 the user enters the URL of the server from a Web client. Since the home URL in the server document for this site defaults to ?Open the user sees a list of databases on the server. Each of these database entries is an HTML anchor, pointing to a Domino-generated URL for the database. The user selects a database by clicking on it, in our example the Tivoli Q&A Database which is tivoli.nsf within Notes. This particular database has no launch options defined so the user is presented with a list of available views. Once again, these views are

HTML anchor tags, in which Domino has placed an OpenView URL using the unique ID of the view. By clicking on the view the user sends a request to Domino to open the specific view. Once in the view, the user is presented with another list of hypertext links to different documents. Domino has constructed the OpenDocument URLs of these links using the unique ID of the view and of the document.

Table 1 shows some examples of the translation Domino performs to change Notes commands into Domino URLs for a server named srv1 and a database named db.nsf.

Table 1. Domino URLs

Notes Command	**Domino URL**
@Command([OpenView];"Discussion"	http://srv1/db.nsf/Discussion?OpenView
@Command([OpenNavigator];"View Selector"	http://srv1/db.nsf/View+Selector?OpenNavigator
@Command([Compose];"Question"	http://srv1/db.nsf/*form UNID*?OpenForm
@URLOpen("https://srv1/db.nsf?OpenDatabase")	https://srv1/db.nsf?OpenDatabase

2.2.2 Coding HTML in Notes

HTML can be coded directly into forms by enclosing the HTML in brackets. For example:

```
[<A HREF=http://www.redbooks.ibm.com>Find out about redbooks</A>]
```

The size and length of fields can be controlled by putting HTML commands in the field Help description box in the Field Properties InfoBox. Figure 7 on page 12 shows an example of both. The titles for the Question and Summary fields have been defined using HTML tags. The Qsum field has been defined to have a default size and maximum length by putting the HTML specifications in the Help Description field.

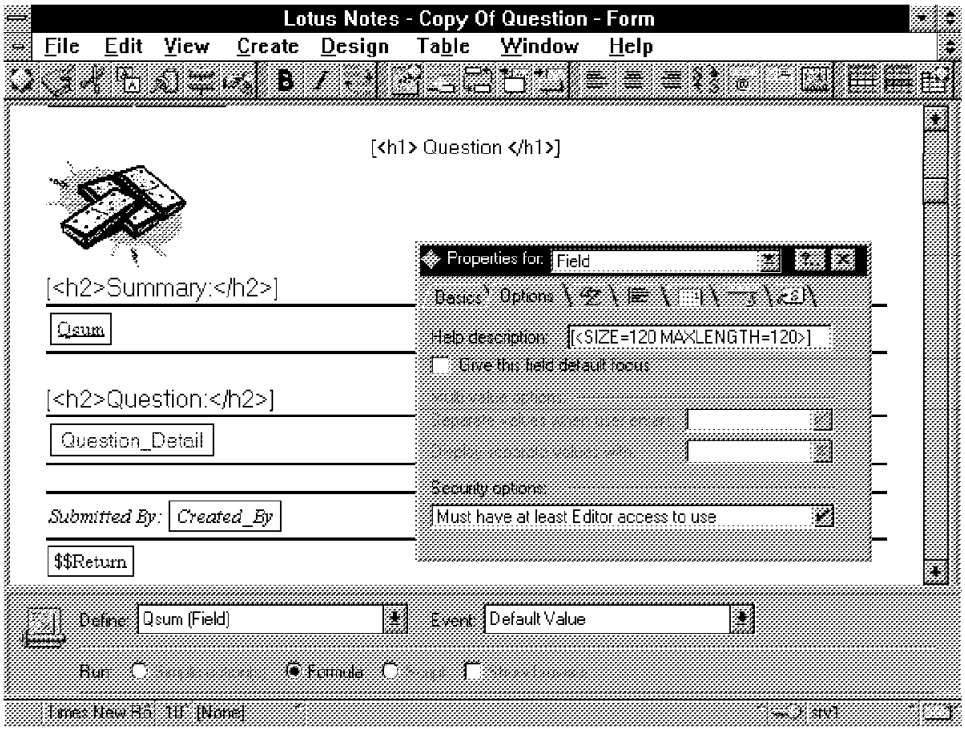


Figure 7. HTML in Forms

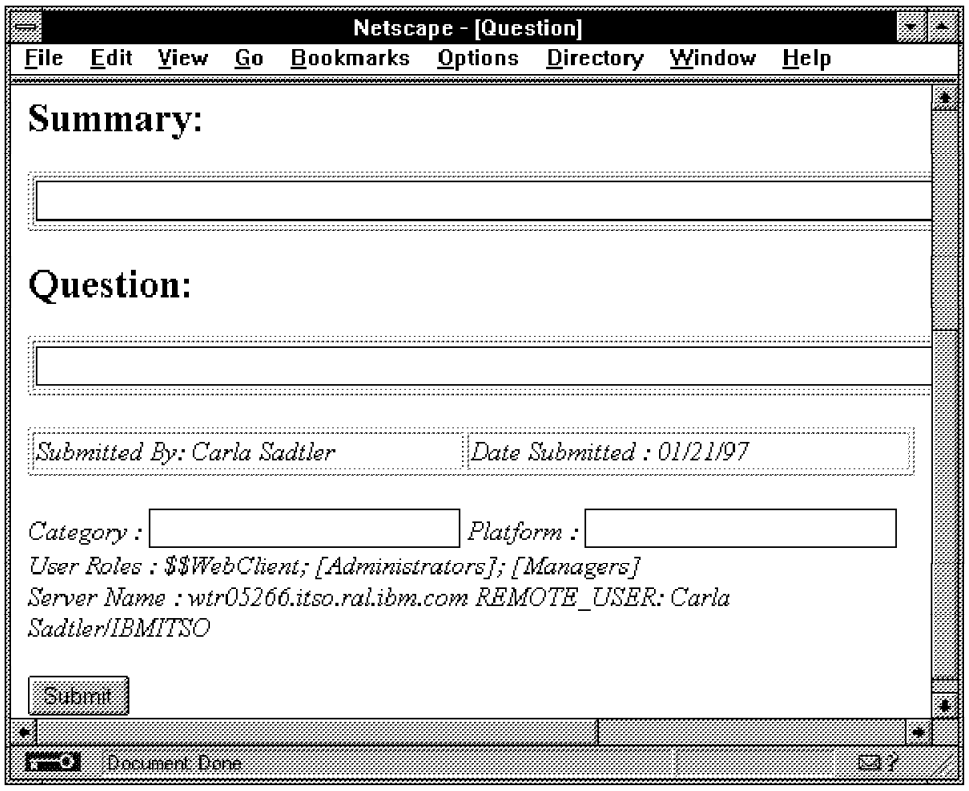


Figure 8. HTML in Forms from Web Browser

2.2.3 CGI Variables

The Common Gateway Interface (CGI) is the most common way to invoke server-side processing in the World Wide Web. CGI may be invoked by the user either by clicking on the submit button in an HTML form or selecting a URL that the server resolves to an executable program. However it is invoked, the interface causes data from the fields in the form or from within the URL to be passed as variables to the CGI program. There are also other CGI variables that are always implicitly set and presented by the browser. These variables usually contain environmental information such as the IP address of the browser.

In a Domino server, submission of a form may result in a standard CGI program being executed, or it may resolve into a Notes function, such as opening a view. In the latter case, Domino gives you access to all of the CGI variables directly. By coding a field in a form with the CGI variable name, the application will automatically have access to the value of the variable. Examples of CGI variables available to Domino applications are Remote_Host, Remote_User, and Server_Name.

Figure 9 shows an example of coding a CGI variable in a document. The Server_Name and Remote_User fields have been added to this form.

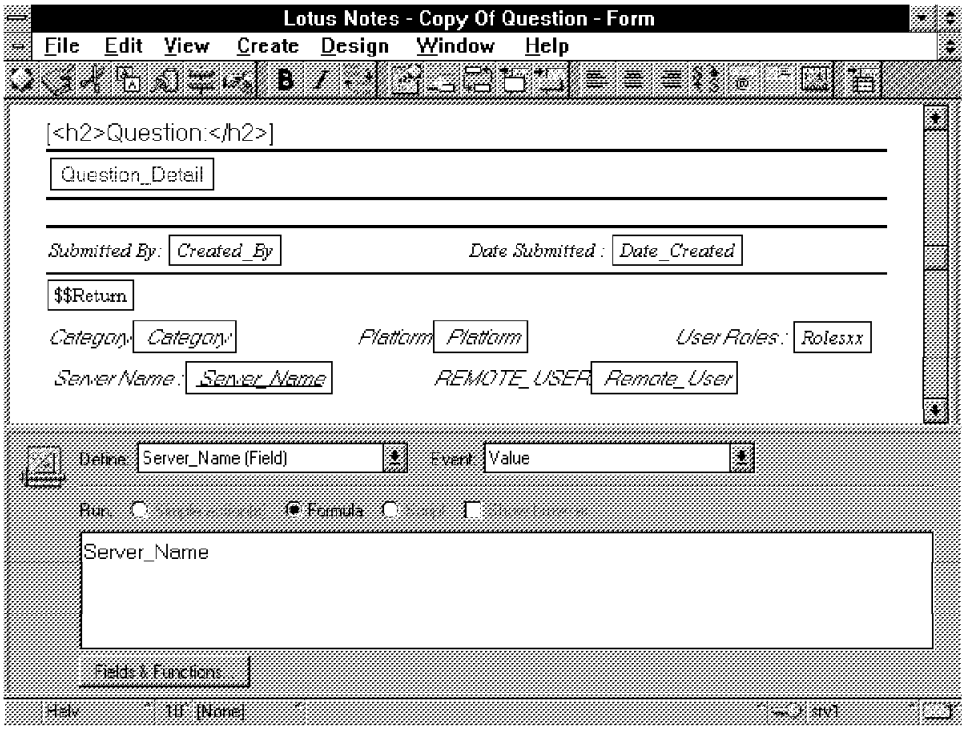


Figure 9. CGI Variables

The effect of coding these CGI variables in a Notes form can be seen in Figure 8 on page 12.

2.3 Domino Security

We have shown in this chapter how the Notes and Web environments are molded together by the Domino server. You have seen how the two different clients have a lot in common and how Domino performs a mapping between them. There are some areas where the hybrid environment places limitations on the application developer, but in general it is possible to take a Notes application and make it available to the great horde of Web browsers with minimal alteration.

But the biggest gap that Domino has to bridge between the Notes and Web environments is in the area of security. In Notes, security is fundamental to every operation. It is not possible to do anything without first establishing your credentials. By contrast, security in the Web is almost an afterthought; something that is incidental to the basic operation of servers and browsers.

Security is what this redbook is all about. In the following chapters we go into the details of Web and Notes security features and explore how Domino pulls them together. However, to round off this description of how Domino works, here we summarize the techniques it uses to map security functions:

- Authentication of Web users is done by using HTTP basic authentication, keeping the user name and password in the Notes Public Address Book. Basic authentication is discussed in 4.1.1, “HTTP Basic Authentication” on page 38 and the Domino implementation is discussed in 5.1, “Access Control and Authentication” on page 51.
- Domino also makes it possible to access a Notes server without first identifying yourself, something that has not been possible before. Web users not in the Public Address Book are considered to have the name Anonymous. The Anonymous user ID is used as a normal Notes Web ID and can be assigned to security roles, groups, and ACLs. Defining Web users is covered in more detail in 5.1.2, “Defining Web Users” on page 53.
- SSL capabilities are supported for connections requiring confidentiality and server authentication. SSL is discussed in more detail in 4.1.2, “Secure Sockets Layer (SSL)” on page 41 and 5.2, “Confidentiality” on page 56.
- Access control facilities in databases are used for security. These facilities are the same as in traditional Notes development with a few exceptions. Database access control facilities are covered in 3.1.3, “Data Access Security” on page 18. The exceptions in a Domino environment are discussed in 5.1.4, “Access Control Limitations” on page 55.

Chapter 3. Lotus Notes Security Features

Notes is designed to be a well-managed, secure environment. It has a range of facilities addressing the security objectives that we set ourselves in 1.1, "Security Objectives" on page 1. This chapter describes various aspects of Notes security, including facilities for access control, authentication, confidentiality, data integrity and non-repudiation.

The base on which all Notes security is built is user authentication. Authentication is very important because it allows you to differentiate one user from another. Without it, all would be treated equally, and you could not identify whether people are who they claim to be. It is therefore the key to providing restricted access to Notes resources. Because of its importance, we explore the methods used by Notes for authentication in some detail in 3.2, "Certification and User IDs" on page 22. Confidentiality and integrity, as they apply to database replication and mail items in transit through the network, ensure that what arrives is identical to what departed, and only the intended recipients can access the contents. Non-repudiation keeps track of who did what, when. All these features taken together constitute the security infrastructure of Notes.

Notes makes use of a number of cryptographic techniques, such as public key and symmetric key encryption, digital signatures and public key certificates . If you want to learn a little more about these things, you may want to read Appendix A, "Cryptographic Techniques" on page 149 before proceeding.

3.1 Notes Access Control Facilities

Access control facilities within the Lotus Notes product itself can be looked at as seven layers from the outside in. The outside layer is simply the security involved in having any access to the Notes structure. The innermost layer involves security at the field level within a Lotus Notes document. Figure 10 on page 16 depicts the seven layers.

© Copyright IBM Corp. 1997

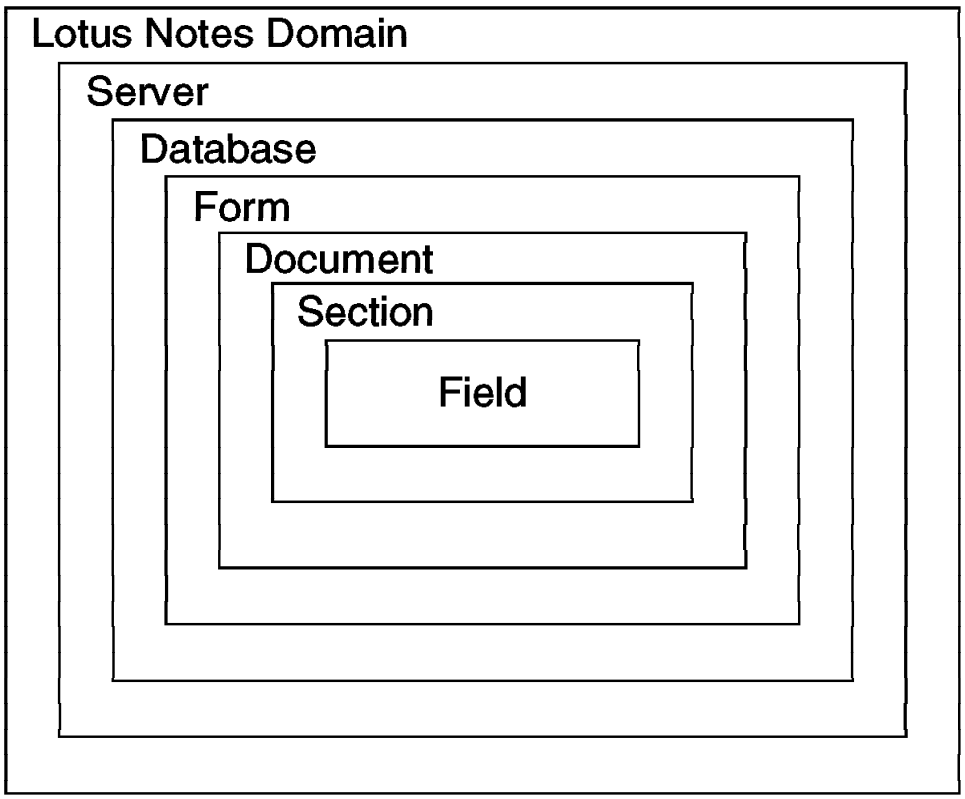


Figure 10. Seven Layers of Lotus Notes Security

We now describe the function of each of these security layers.

3.1.1 Domain Security

In Notes, a *domain* is a group of servers and clients that fall under the same administrative structure and therefore share some key configuration information, including user and group definitions. This domain-wide information is maintained in the Public Address Book. The security of a Notes domain includes both physical and logical access to the servers, workstations, and certificates.

The groups and users registered in the domain have different levels to access the Notes servers and databases. Special people such as certification administrators, server administrators and database managers who have higher priority access to the servers and databases, play an important role in security. Delegation of these responsibilities should be done with care.

Servers store important databases and should be treated as precious resources. If they are destroyed or corrupted, the loss can be a serious blow and recovery may be very costly in terms of time and money, if possible at all.

In order to prevent disaster from brute-force attacks, the Notes servers should be in a physically secure and environmentally sound area. Every Notes server and workstation should have passwords to restrict administrator access. User access to databases and servers should be carefully controlled.

3.1.2 Server Security

Server documents in the Public Address Book are used to control access to a Notes server. The implementation of security does not come for free, of course. Applying server access restrictions will activate an additional security code that uses server processing cycles and can increase the time taken for a user to gain access the server. They are, however, an effective place to start with security controls. Using the restriction fields in the server documents you can quickly restrict several types of access to the server. You update the server access control parameters by selecting **File** and then **Database** from the menu bar and then selecting the **Public Address Book**. The server document can be selected from the navigator portion of the screen. Figure 11 shows an example of a server document and Table 2 details the controls available.

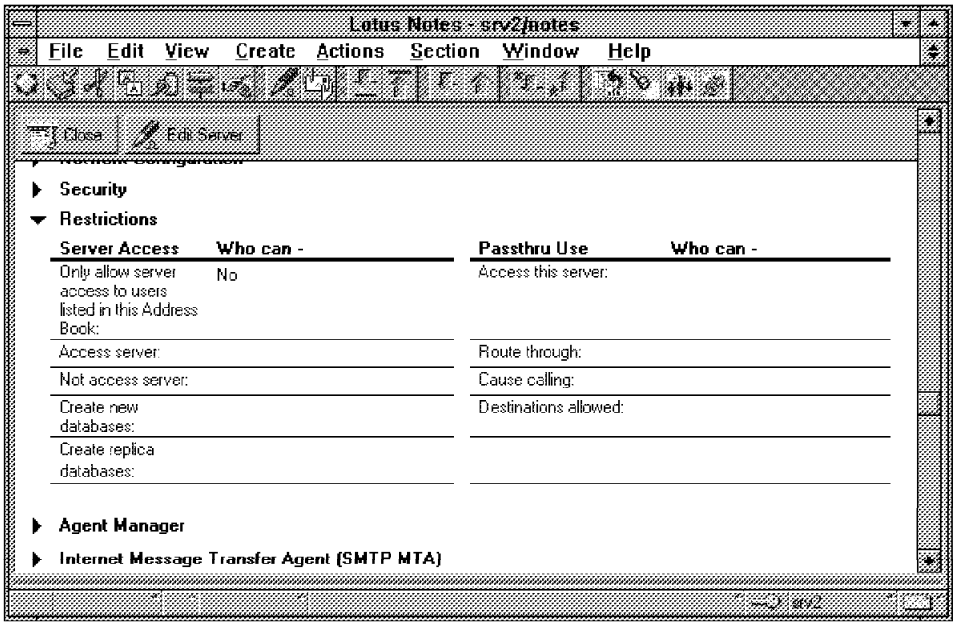


Figure 11. Restriction Section in Server Document

Table 2 (Page 1 of 2). Server Restriction Controls

Server Document Field	Purpose
Only allow server access to users listed in this Address Book	Choose Yes if you want only those people who have person records to be allowed to access this server.
Access server	People, servers, and groups allowed to access server. No entry allows access to all.
Not access server	People, servers, and groups *not* allowed to access server. No entry denies no one.
Create new databases	People, servers, and groups allowed to create new databases on the server. No entry allows all.
Create replica databases	People, servers, and groups allowed to create replica databases on the server. No entry allows no one.
Access this server (Passthru Use)	People, servers, and groups allowed to reach this server via passthru routing. No entry allows no one.
Route through (Passthru Use)	People, servers, and groups allowed to route through this server. No entry allows no one.

Table 2 (Page 2 of 2). Server Restriction Controls	
Server Document Field	**Purpose**
Cause calling (Passthru Use)	People, servers, and groups allowed to ″call″ other servers while routing through this server. No entry allows no one.
Destinations allowed (Passthru Use)	Remote servers which this server may route to via Passthru. No entry means all servers may be routed to

3.1.3 Data Access Security

After users or other servers gain access to a server, they will want some level of access to the data held by that server. The remaining security layers within the Notes security model (see Figure 10 on page 16) deal with data access, each layer providing more granular control than its predecessor. In fact, the access controls mirror the way that Notes stores and presents data:

Database access At the heart of the system lie Notes databases. The records in a database are actually documents that have usually been entered by a user or administrator. Database access control facilities therefore provide the broadest control over who can do what to data on a Notes server.

Form access Documents are not free-form text, but are in fact filled-in *forms*. When designing forms you can use form access controls to specify who has access to the contents of a database in more detail than using the database access controls.

Document access Once a form has been filled in to create a document, the owner of the document can further restrict access to it. To what degree this is allowed depends on controls within the form from which the document is created.

Section access Many documents contain data of varying sensitivity. In practice this means that you want to prevent certain users from updating or possibly even from reading parts of the document, but you want other users to have full access. One way to achieve this is to divide the form into sections and apply section access controls to it.

Field access This is the most granular form of data access control. It allows you to control access to individual fields on a form or document. In addition to specifying user access, field access controls can limit the treatment that data receives when it is transmitted or stored.

We now consider each of these access control layers in more detail.

3.1.3.1 Database Access Control

Every database has an access control list (ACL) that controls user, server, and group access to the database. For example, you can create an ACL that denies database access to some users, allows other users to read documents in the database, and allows yet other users to change the design of the database.

The access levels, from highest to lowest, as they apply to users are as follows:

Manager Users with manager access can modify ACL settings, encrypt a database for local security, modify replication settings, and delete a database, tasks permitted by no other access level. Managers can also perform all tasks allowed by other access levels. Notes requires each database to have at lease one manager. It′s best to assign two people manager access to a database in case one manager is absent.

Designer Users with designer access can modify all database design elements (fields, forms, views, public agents, the database icon, Using This Database document and About This Database document). They can also modify replication formulas, and can create a full-text index. Designers can perform all tasks allowed by lower access levels. Assign designer access to the original designer of a database or to a user responsible for updating the design after a database is in use.

Editor Users assigned editor access can create documents and edit all documents, including those created by others. Assign editor access, for example, to a user responsible for maintaining all data in a database.

Author Users assigned author access can create documents and edit documents they create. Assign author access to allow users to contribute to a database but not edit documents created by others.

Reader Users assigned reader access can read documents in a database but cannot create or edit documents. For example, assign reader access to users who must be able to read the contents of a reference database such as a company policies database.

Anyone with at least reader access to a database can create personal agents in the database if the database manager selects the ACL option Create personal agents. However, users can only run agents that perform tasks allowed by their access levels. For example, someone with reader access can create a private agent that deletes documents, but the agent won′t delete documents when the user runs it.

Depositor Users assigned depositor access can create documents but can′t see any documents in the database views, even the documents they create. For example, assign depositor access to allow users to contribute to a mail-in database or to a database used as a ballot box.

No Access Users assigned no access can not access the database. For example, using no access as the default prevents most users from reading a confidential database.

Note: Users with no access or depositor access to a database may be able to copy the database outside of Notes through the operating system. Notes administrators can prevent this by keeping the Notes server physically secure. A database manager can also encrypt a database with the server ID through the local security option so that the database, even when copied, is illegible to anyone who doesn′t have access to the server ID. If you encrypt a database with a server ID, make sure the server ID is secure.

3.1.3.2 Form Access Control

Default read and create access to forms can be specified at the form level using the security section of the forms property box. Using the forms security will override the database ACL, allowing more flexibility in database security designs. You can restrict people who have read access to the database from reading documents created with the form. In addition you can restrict people from creating documents with this form even though they have author (or above) access to the database.

Table 3. Security Options in Form Properties Box

Option	Use
Default read access for documents created with this form	Allow only a subset of users in the database ACL to read documents created with a specific form. Documents created with this form will have this subset of users as the default document reader access list.
Who can create documents with this form	Allow only a subset of users with author access or above in a database ACL to use this form to create documents.
Default encryption keys	Encryption keys are generally created by database managers and distributed to the appropriate users. Specifying this will cause all encryptable fields to be encrypted when a document is saved with this form.
Disable printing/forwarding/copying to clipboard	This feature is an aid to users to keep them from accidentally including sensitive data when reproducing a document. It is not a very secure measure so it should not be relied on for true security.

3.1.3.3 Document Access Control

The creator of a document can determine who can read the document by using the security section of the document properties box. They can choose to allow all people with reader access or above to read the document or restrict read access to a limited number of people.

Documents inherit their read access property from the read access property in the form used to create the document. Anyone allowed to edit the document can change the document read access property.

3.1.3.4 Controlling Access to Documents Using Field-Based Access Controls

Each document can have special fields defined within it that control access to it. These fields are reader fields, author fields and signed fields.

Reader fields Reader fields can be created on a form to allow read access restrictions to documents. A reader field consists of a list of names to be allowed read access to the document. It can be used instead of or in conjunction with form and document security read access lists. If both read access lists and reader fields are present the users who can read the document are the addition of both lists. A reader field can not allow access to a user that does not have read access to the database. It can, however, prevent read access from someone in the database ACL that could normally read the document.

Designating a field as a reader field and as editable allows the designer to choose options for presenting name lists to to the author of the document. The field could present a

list of names from the address dialogs, database ACL, a view dialog, or no list so the author would manually enter names.

A useful way to use a reader field would be to create the field so the author of a document is presented with a list of people or groups to choose from, allowing the author to determine the reader list on a document-by-document basis.

Author fields Author fields can be created on a form to give users with author access edit capability to a document they didn't create. Author fields contain names of users and are created in the same way as reader fields. It is only applicable to users with author access to the database.

Signed fields Signed fields can be created on a form to allow a digital signature to be attached when a document is saved or mailed. Digital signatures verify that authors are who they say they are and guarantee that the data in the document has not been tampered with. The private key in a user ID file generates the signature. When a user opens the document containing the signed field, Notes verifies the signature by comparing it with the author's public key in the Public Address Book.

3.1.3.5 Restricting Access to Sections within a Document

Standard sections are used in a form to collapse or expand information. They can be hidden based on whether a document is in read or edit mode, or based on a formula. This will only hide a section from view. It does not protect it from update from agents, actions, or access from another form.

Access-controlled sections are used to group areas of a form and control edit access to objects in that area. However, like the standard sections, these fields may be edited from other forms, actions, or agents.

Layout regions are similar to sections. They are areas of grouped objects that can be easily moved and displayed in ways not available with forms and subforms. As with sections, they can be hidden based on whether a document is in read or edit mode, or based on a formula. This will only hide a layout region from view. It does not protect it from update from agents, actions, or access from another form.

3.1.4 Field Access Control

Database designers can design fields that to be encrypted with an encryption key. To decrypt and read the document, users must have the same key.

Fields may also be protected during form design from update by authors after the initial document is created. Field property security options include an option specifying that a user must have at least editor access to use the field.

3.2 Certification and User IDs

Notes security is based on public-key certificates, contained in user IDs. User IDs are binary files owned by the user. The ID file contains user-specific information such as the user's name, plus certificates and encryption keys.

Certificates are actually signed messages added by a certifier to a user ID file. The certificate contains a number of things:

- The certificate owner's name and details
- The certificate owner's public key
- The certifier's name and details
- The certifier's public key
- Certificate expiration date

The whole thing is then digitally signed by the certifier, to prove its authenticity. Notes certification and authentication is a fairly complex process. We give you a simplified overview of the process below.

3.2.1 Certification

Notes provides two types of certification, flat and hierarchical. Organizations using flat names may use several certifier IDs. Each user ID and server ID can include separate certificates generated by each flat certifier ID. Organizations using hierarchical certification have one organization certifier and optionally up to 4 layers of organizational unit certifiers below.

Flat names are supported for compatibility with earlier versions of Notes. New installations are encouraged to start with hierarchical names, and existing flat installations are encouraged to convert to hierarchical.

3.2.1.1 Hierarchical Certification and User Names

In hierarchical certification an organization may be layered with the organization certifier at the top and up to four layers of organizational unit certifiers below. Users in the organization have fully distinguished names based on their certifiers. Each layer in the certification hierarchy inherits the fully distinguished name of the certifier used to create it and is in turn an ancestor to the layers below it.

When users or servers are registered with a certifier they receive a certificate signed by that certifier and inherit the certification hierarchy of the layers above. For example, consider the certification hierarchy shown in Figure 12 on page 23. This shows an organization named *WidgitCo*, subdivided into two organizational units, *East* and *West*. When Bob Smith shows up on his first day of work, the administrator of East/WidgitCo defines him as a new user. One of the results of this process is a new, randomly-generated, public key pair. The administrator then creates a certificate for Bob, by signing his new public key using the East/WidgitCo certifier private key. As a result, Bob Smith's user ID inherits the certification hierarchy of the East/WidgitCo certifier. Users may authenticate with each other if they have at least one common ancestral certificate. In our example, this means that all users in the organization in Figure 12 on page 23 can authenticate with each other because they have the WidgitCo certifier in common. Entities that don't share at least one common ancestor can still authenticate by going through a cross-certification process.

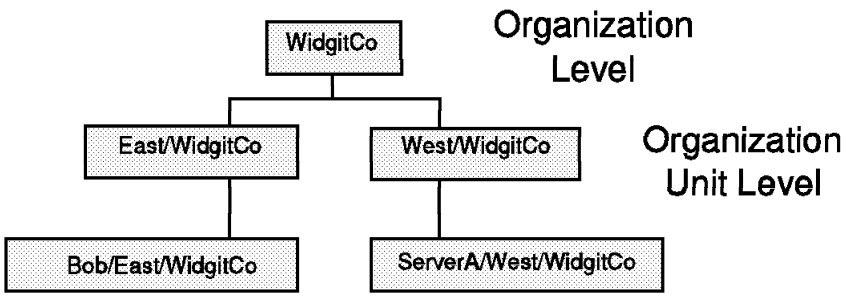


Figure 12. Hierarchical Organization

3.2.2 User ID Contents

Now let us see how this certification hierarchy is reflected in the Notes user ID files and Public Address Book. When an administrator registers a user he or she specifies the user name, password, expiration date, and other default options. The registration process creates an ID for the user or server and places it in the Public Address Book or in a file which must be given to the user to reside on the user's workstation.

Notes generates two RSA key pairs for the user or server. One pair is used for international mail. The other is used for US mail, all electronic signatures and authentication. Notes builds a certificate and uses its private key to sign the certificate. The certificate is then placed in the ID file. For more information on key encryption refer to A.2, "Public Key Encryption" on page 151.

After the registration process the ID file contains:

- The user's name and Notes license number
- Two public and private key pairs
- Two certificates for the user
- A certificate for each ancestor certifier

After registration, encryption keys distributed by application developers may be added to allow encryption and decryption of fields in documents. The private key and the encryption keys in the ID file are encrypted using a key computed from the user's password, so that only the owner can use it. Public information such as the user's name and public key are not encrypted.

Again using the organization shown in Figure 12 as an example, Figure 13 on page 24 shows the ID files for user Bob Smith in the East organization and a server in the West organization.

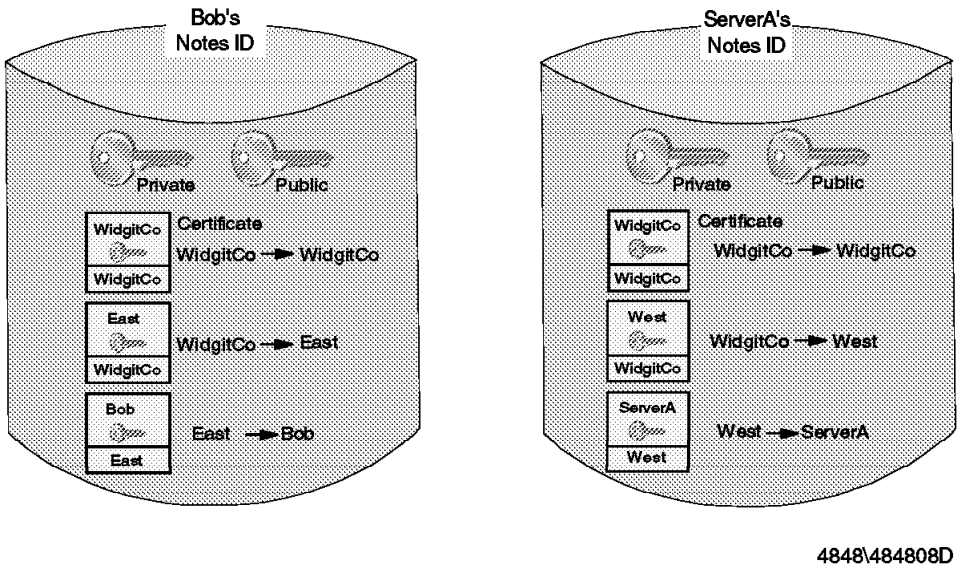


Figure 13. User ID File Contents

3.2.2.1 Displaying the Contents of an ID File

Users can check the certificates available in their ID file by selecting **File**, then **Tools** and then **User ID** from the Notes client menu. Figure 14 on page 25 shows one of the certificates from the ID file for Bob Smith (above). Notice that the entries in the list at the top of the panel show the certifier, not the owner of the certificate. The entry labelled /WidgitCo is the certificate of the East certifier, issued by the top-level certifier in the domain. The two entries labelled /East/WidgitCo are two certificates for Bob Smith, both signed by the East certifier. The reason there are two is that one is an international key with a restricted key size, whereas the other is a full-strength North American key. We discuss export restrictions in 3.4, “Facilities for Confidentiality” on page 30.

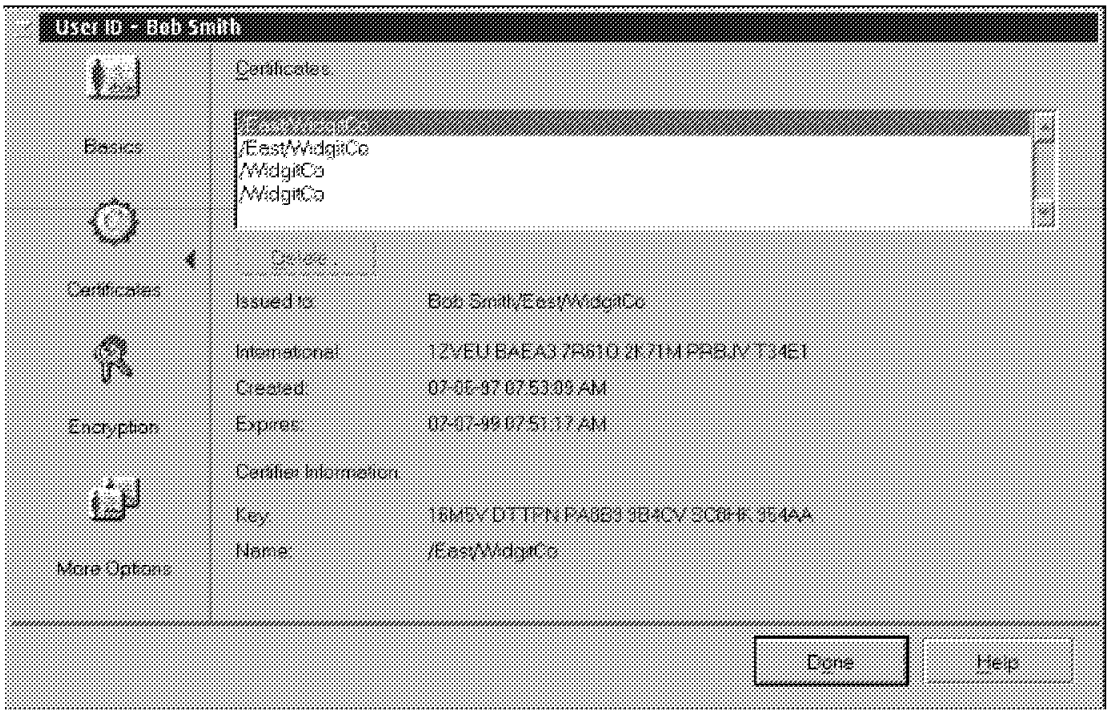


Figure 14. User ID File Contents

3.2.3 User Passwords

The password assigned to a user during registration is a mechanism to protect access to the ID file. A Notes user attempting to use the ID file will be required to enter the password for that ID file. The password is normally used only to unlock the user ID file itself; it is the key pair contained in the file that is actually used to identify the user. A user may have more than one copy of his ID file and different copies can have different passwords. Note that this means that to change the password you must know the existing password. If the ID file is destroyed or lost, the password is useless. This also means that if a user forgets his or her password, he or she must be re-registered and a new ID file created. Backing up ID files and passwords is strongly suggested.

Notes 4.5 adds a password checking process to the server. When password checking is enabled, information dependent on the user′s password and the date the password was provided is kept on the server in the Person document. The user must have entered the password corresponding to the information stored in the Person document to gain access to the server. The password checking facility adds the capability to require user password change intervals and to keep the previous 50 old passwords from being re-used. Notes is using the RSA key pair for authentication, so even if someone guessed the user′s password they would still need to steal the user ID file to be able to impersonate the user. The information stored in the Public Address Book is not subject to dictionary attacks unless the attacker also has the ID file.

3.2.4 The Public Address Book

Information about each ID is also maintained on the Notes server. The Public Address Book contains a Person document for each user, with a lot of information about the user including:

- The user's name and domain
- The user's public key
- An attachment with the user's ID file to be distributed the first time the user contacts the server (if it was stored here instead of in a file)

If a server has been registered, a Server document is put in the Public Address Book with similar information about the server. Certifiers are also represented in the Public Address Book by Server Certificate documents. Figure 15 shows the Name and Address book entries for our sample configuration.

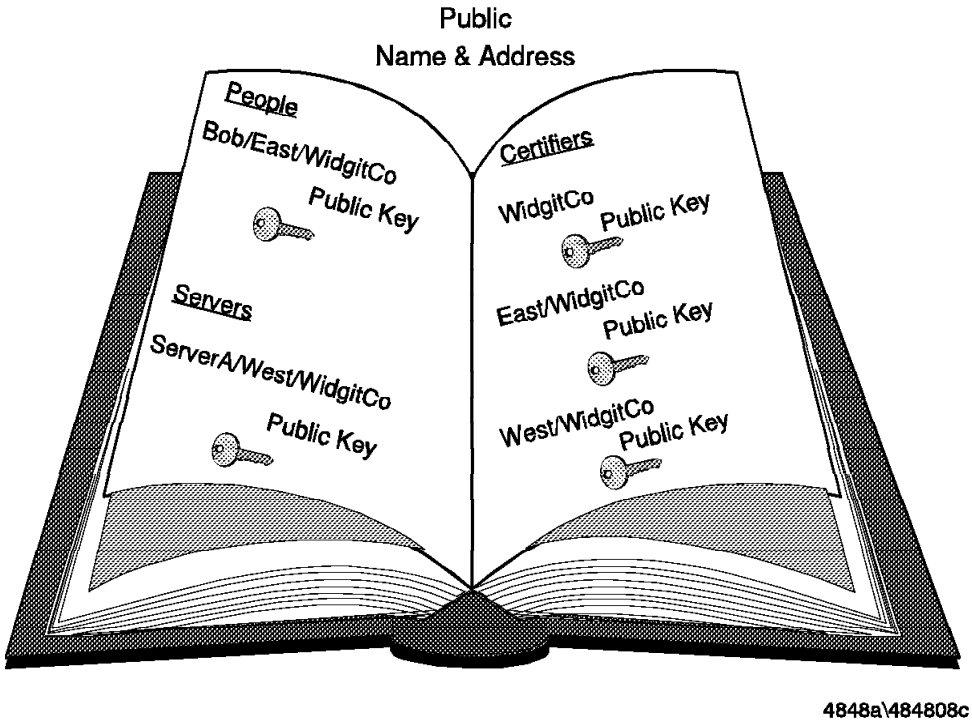


Figure 15. Public Address Book

3.2.5 Authentication and Validation

Verifying a user's or server's identity in Notes includes two steps. The first step, *validation*, is the process of reliably determining the sender's public key. Notes uses the following three rules when deciding to trust a public key:

1. Trust the public key of any of your ancestors in the hierarchical name tree because they are stored in your ID file.
2. Trust any public key obtained from a valid certificate issued by any of your ancestors in the hierarchical name tree.

3. Trust any public key certified by any trusted certifier and belonging to one of the certifier's descendants.

Let us now see how these rules are applied in the validation process. The user ID file for Bob Smith contains everything he needs to identify himself and establish his credentials. When he requests a session with a server the first step is to send to the server all of the certificates from the ID file (both the user's own certificate and the chain of certifiers' certificates that support it). Figure 16 illustrates the validation process that follows.

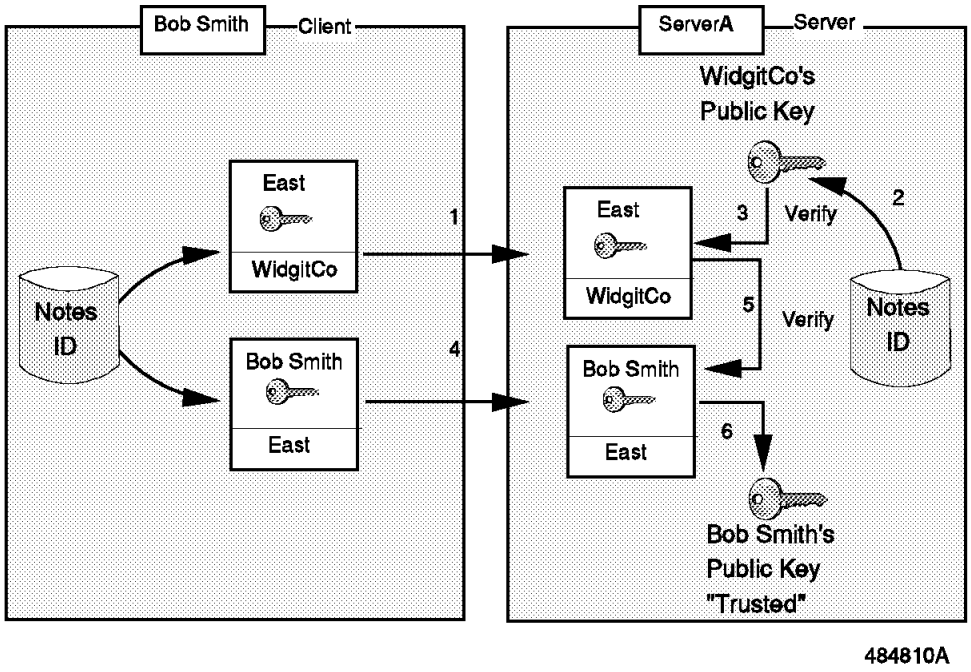


Figure 16. Validation Process

The steps shown numbered in the diagram are as follows:

1. ServerA reads the East certificate that Bob Smith sent from its ID file. This was signed by WidgitCo. ServerA is interested in it because East is the certifier of Bob's certificate.
2. ServerA reads the WidgitCo public key from its own ID file. (According to rule 1, ServerA will trust the public key of any ancestor that is stored in its ID file.)
3. ServerA uses the public key of WidgitCo (which is trusted because it is in the server's ID file) to verify that the certificate of East/WidgitCo is valid. (According to rule 2, if you trust the public key of the ancestor, you will trust any public key obtained from certificates issued by the ancestor.)
4. ServerA reads Bob Smith's certificate that was sent from his ID file. This was signed by East.
5. ServerA uses the public key of East/WidgitCo, which now is trusted, to verify that the Bob Smith/East/WidgitCo certificate is valid. (According to rule 3, trust any public key certified by any trusted certifier and belonging to one of the certifier's descendants.

6. ServerA has now reliably learned Bob Smith's public key.

   The same process is followed in reverse so that Bob can reliably learn ServerA's public key.

   After the validation process finishes the authentication process begins. *Authentication* is a proof of identity. You may think that the validation process (above) has proved who each of the session partners is. Not so, because all they have presented so far is certificates. A certificate associates the user with a public key and tells the recipient that the public key can be trusted, but in order to prove that users really are who they claim to be they must show that they hold the private key that matches the public key in the certificate. The authentication process achieves this with a challenge/response dialog between a workstation and a server, or between two servers when either is running database replication or mail routing.

   To continue the previous example of a Bob Smith accessing ServerA, see Figure 17. The following is an over simplification of the actual process and is intended to illustrate what happens in a manner that's easy to understand.

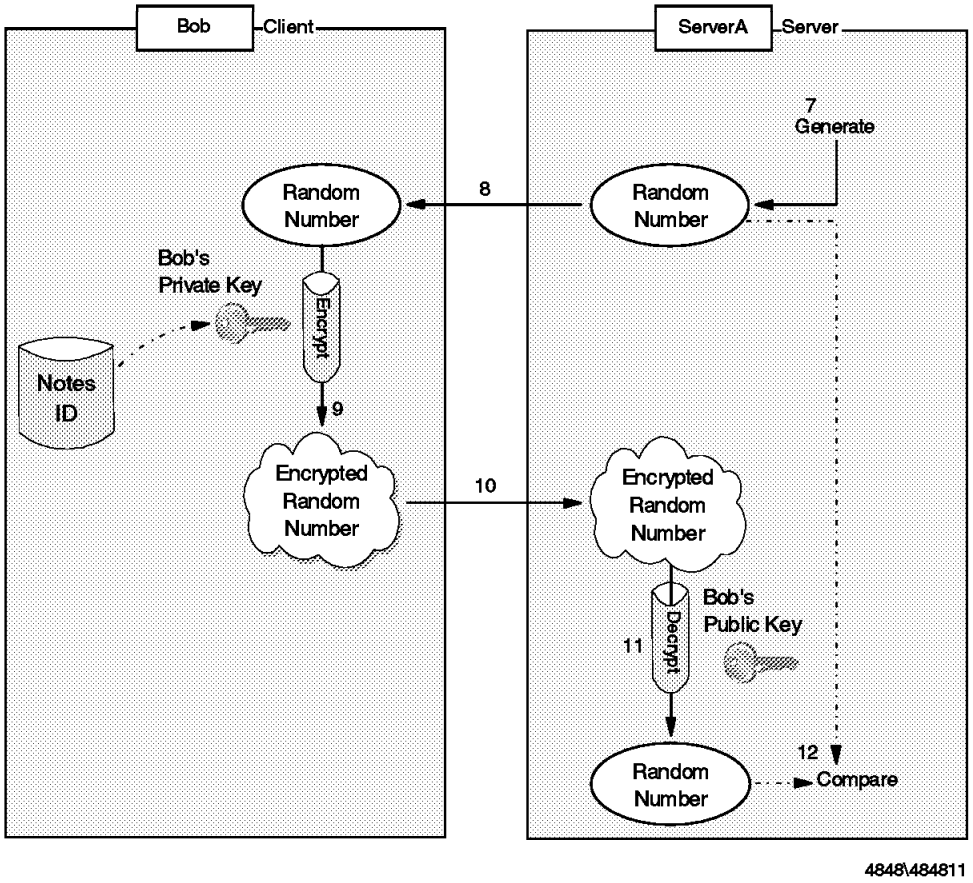


Figure 17. Authentication

7. ServerA generates a random number and a session key and encrypts both with Bob's public key.
8. ServerA sends the encrypted random number to Bob Smith.
9. Bob receives the challenge and decrypts it with his private key.
10. Bob Smith sends back the decrypted number to ServerA.

11. ServerA compares Bob′s response to the original random number.
12. If the result is the same as the original random number, ServerA can trust that Bob Smith really is who he claims to be.

As with validation, authentication is a two-way procedure. Bob now authenticates ServerA using the same challenge/response process in reverse.

The actual algorithm is complex but efficient. It avoids any RSA operations on subsequent authentications between the same client/server pair. It also establishes a session key that can be used to optimally encrypt the messages that follow authentication.

3.3 Facilities for Data Integrity

When databases are replicated and mail is routed through the network there is the risk they could be modified. We must be able to tell if the data we receive is the same as the original data sent.

In order to detect any changes, we use digital signatures. Data integrity implies the current condition of the data is equal to the original ″pure″ one. It guarantees that information is not changed in transit. A digital signature can verify that the person who originated the data is the author and that no one has tampered with the data. Senders can add their digital signature to mail or to fields and sections of documents to prove that they are the origin.

Note: A database designer controls whether or not fields and sections of a database are signable; individual users can choose to sign mail messages.

Digital signatures use the same RSA key pair that was used in the verification and authentication process (above). Refer to A.4, “Combinations of Cryptographic Techniques” on page 152 for an introduction to digital signatures.

Figure 18 illustrates how Notes uses electronic signatures.

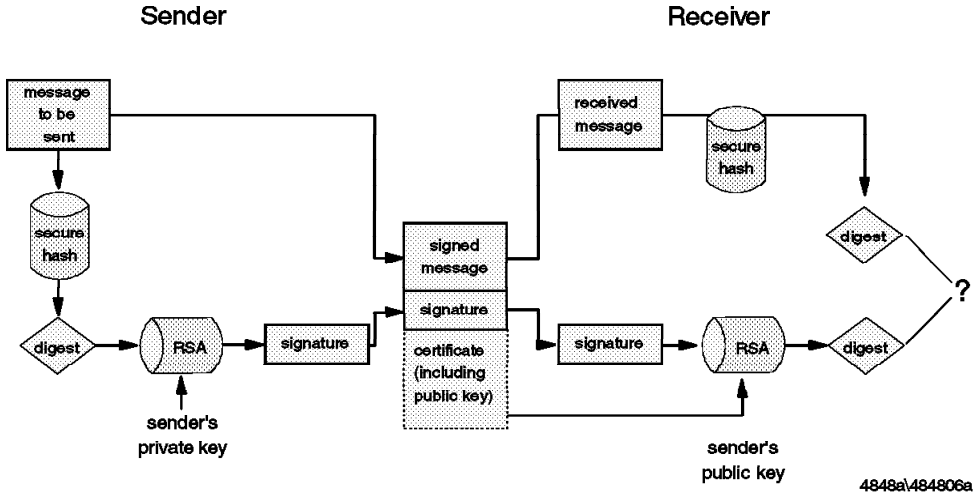


Figure 18. Data Integrity Using Electronic Signatures

When a user signs mail, Notes uses a secure hash algorithm to generate a 128-bit digest of the data being signed and encrypts the digest with the author′s

private key. This signature is attached to the data, along with the signer′s public key and certificates.

When the receiver accesses the signed data, Notes authenticates the sender′s identity and decrypts the signed data using the public key in the user′s certificate.

Notes indicates who signed the message if decryption of the signature is successful. Otherwise, Notes indicates that it can not verify the signature. Two things are guaranteed by this signature process: the sender is authenticated (because the digest must have been encrypted in the sender′s private key), and the message arrived unmodified (because the digests are identical). Otherwise the receiver knows the data has been tampered with or that the sender does not have a certificate trusted by the reader.

Note: Because Notes must authenticate the sender before attempting to decrypt the signature, this means that electronic signatures only work for users who can authenticate. You will not be able to verify a signature from users in other organizations that are not cross-certified.

3.4 Facilities for Confidentiality

When you send data through the network, including mail, anyone who can intercept network packets (by tracing or electronic sniffing techniques) may read your data without authentication. You may not care if the information is not private. But you may care that every piece of mail you send or receive is possibly being read by others. This lack of privacy is a serious problem: it is likely that 90% of mail traffic is not sensitive, but to solve the problem you either have to persuade users to take security seriously or you have to treat *all* mail as sensitive and encrypt everything. Experience shows that upgrading computer systems is easier than modifying human nature, so often the latter approach is applied.

Fortunately, Notes provides a solution for either approach. Sensitive data can be encrypted into an unreadable format before transit. After it arrives at the destination, decryption can be used to read it. This method protects data from unauthorized access. Notes uses a bulk encryption mechanism, based on a secret key to encrypt and decrypt data and to confirm that the data you received hasn′t been read by others.

After authentication, the client begins to exchange data with the server. A lot of data will pass through the network, so it is important that the algorithm used is efficient. Notes uses the RC2 or RC4 algorithms for bulk encryption of data. (See A.1, “Symmetric Key Encryption” on page 149 for information about bulk encryption and the RCx algorithms).

One variation is introduced by the type of Notes license a user has. There are two types of licenses in Notes, North American and International. North American and International licenses are compatible in almost all aspects. For example, certification, authentication, electronic signatures, and communication between servers and users, is the same except for one significant difference.

Because of U.S. government export restrictions on encryption technology, a North American license uses a more secure encryption key than an International license. Notes will automatically figure out whether the sender or any of the recipients of a message is international and if so will use the less secure key.

Figure 19 on page 31 illustrates the difference between a North American and International license.

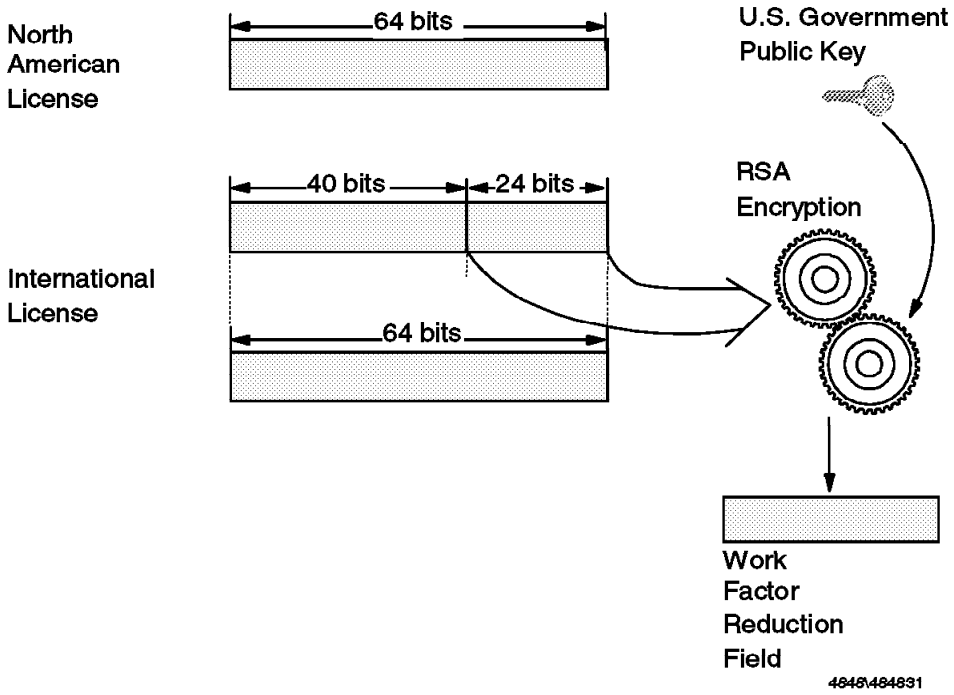


Figure 19. The Difference between a North American and International License

No matter which version of Notes you are using, encryption uses the full 64-bit key size. However, the International edition takes 24 bits of the key and encrypts it using an RSA public key for which the U.S. government holds the matching private key. This encrypted portion of the key is then sent with each message as an additional field, the *workfactor reduction field*. The net result of this is that attackers must break 64-bit encryption, which is at or beyond the practical limit for current decryption technology and hardware. The U.S. government, on the other hand, only has to break a 40-bit key space, which is much easier (2 to the power of 24 times easier, to be precise).

Notes′ unique encryption technique allows the International edition of Notes to use an encryption key equal in strength to the 64-bit key in the North American edition. This significantly increases international customers′ information security, while neither increasing nor decreasing government access to encrypted information. The two editions of Notes are fully interoperable.

Remember two things. If a user is no longer able to use his or her ID file, either because he or she forgot his or her password or physically lost the file, any mail encrypted under his or her public key is permanently lost. The private key is located in the ID file. Also anybody who has a copy of your private key can read your encrypted mail. Protect your private key carefully.

3.4.1 Mail Encryption Example

The following shows how Notes uses secret keys to encrypt and decrypt mail messages:

1. When the sender encrypts a message, Notes generates a random encryption key and encrypts the message with it. The random encryption key is encrypted with the recipients public key and attached to the message. The RC2 algorithm is used to encrypt and decrypt the data.
2. After the mail with the encrypted key arrives, the recipient uses his private key to decrypt it. Secrecy is guaranteed, because only the recipient's private key can be used to decrypt the secret key needed to decrypt the message.
3. A different random key is generated and sent each time mail is sent.

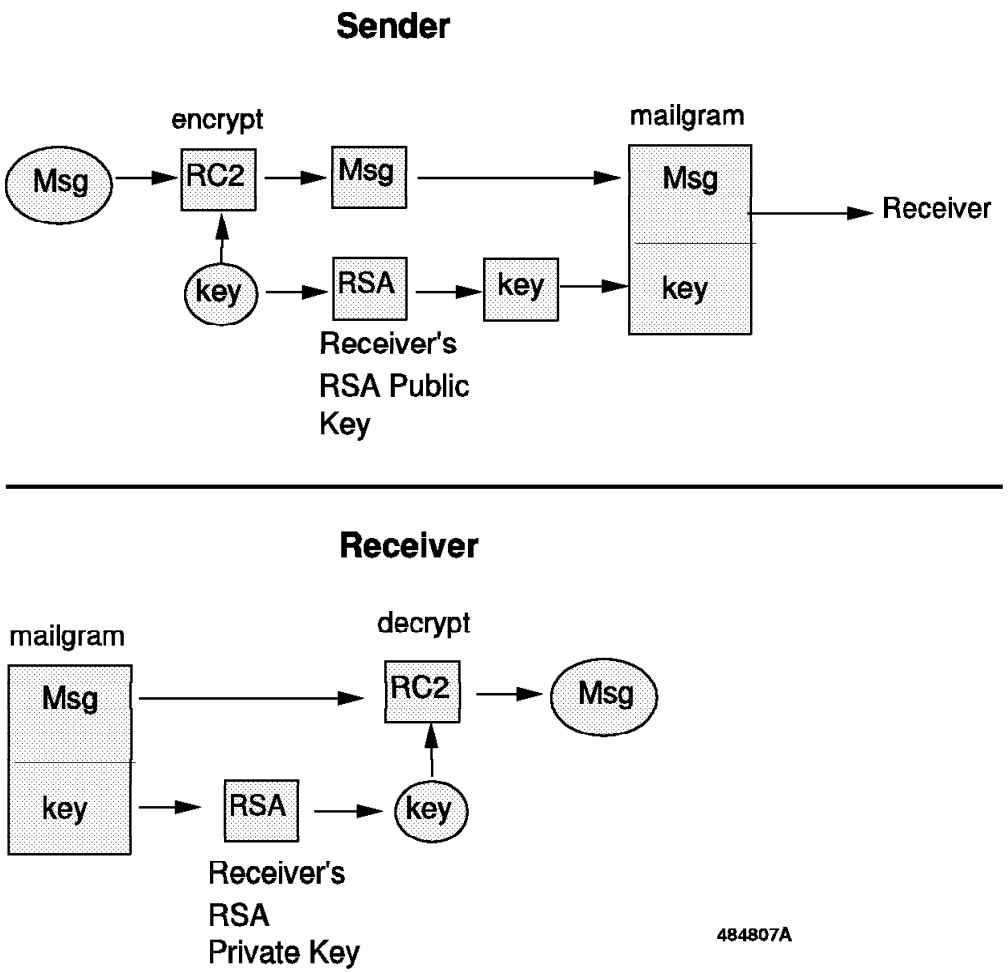


Figure 20. Use the Secret Key to Encrypt and Decrypt the Messages

3.4.2 Other Notes Encryption Features

The above description applies to Notes mail, but Notes also provides other methods for encrypting information. Databases, documents, fields, and transmission of data over the network can be protected using various methods of encryption:

- Databases can be encrypted with a user or server ID by using the Local Security option. This protects the database from being accessed from an unauthorized user who has gained access to the workstation the database is stored on or who has made a copy of the database by using the operating system.
- Field encryption using special encryption keys created and distributed by the database designer can be used to limit access of fields to authorized users.
- Documents can be encrypted using private or public keys. Keys can be added to the form causing every document created with the form to be encrypted, or by letting users encrypt documents with their own encryption keys.
- Network port encryption allows unencrypted data to be encrypted at the port level for safe transport through the network. Network port encryption can be enabled for a user's workstation or at a server by clicking on **Files** → **Tools** → **User preferences** to modify the port definition to encrypt network data.

3.5 Non-Repudiation

Non-repudiation provides unforgeable evidence that a specific action occurred. The service includes proof with authority of the origin, delivery, and submission of information.

Non-repudiation of origin protects against any attempt by a message originator to deny sending a message. Non-repudiation of submission protects against any attempt to deny that a message was submitted. Non-repudiation of delivery protects against any attempt by a message recipient to deny receiving a message. As a whole, it keeps track of all these actions, verifying who did them and the results.

Notes does not provide this kind of service, although it does provide facilities such as logging that may be used to create non-repudiable transactions. You must design an application to provide this if you need it.

For example, digital signatures can be used to provide non-repudiation of origin (refer to 3.3, "Facilities for Data Integrity" on page 29). If a recipient returns a message by signing a report, non-repudiation of delivery is provided. Since only the recipient's private key could have generated the signature, the signature provides unforgeable evidence of message delivery.

3.6 Execution Control Lists (ECLs)

Notes provides a form of workstation security through the use of Execution Control Lists (ECLs). The ECL determines what effect actions in another user's formulas or scripts will have when run on a workstation.

The Notes administrator sets up an ECL on the server in the Public Address Book. The ECL lists users or groups of users and the access they have to a

workstation. During Notes setup for clients this ECL is copied to the user's workstation as a default and is used when a formula or script runs on that workstation. The signature on the code is compared to the workstation's ECL to determine if the action can take place.

Executable code (found in forms, buttons, actions, agents, etc.) is signed with the ID of the user that created it or last modified it. The signature of a database or template can be seen on the status bar when a form is open for design. Templates and databases can be signed with a particular ID by using the SIGNNSF.EXE program included in Notes Domino 4.51. All Notes templates are signed by Lotus Notes Development/Lotus Notes.

The user can update the ECL on his workstation by clicking on **Files** → **Tools** → **User preferences** and then clicking on Security Options. In the following figure you can see that we have turned off the authority to access environment variables from any code signed by Developer One. The last entry in the list is to allow executable code included in templates and databases created by Lotus developers.

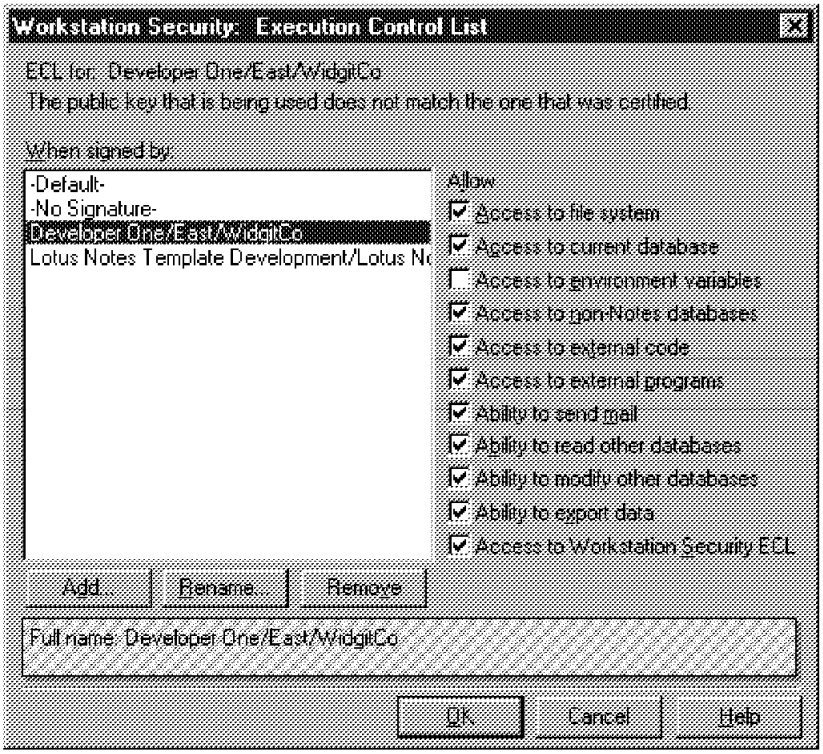


Figure 21. Execution Control List

Executing a test agent that modifies the user's environment variable illustrates the effect of the ECL setting. The following agent changes an environment variable called "testvalue" in the user's NOTES.INI file.

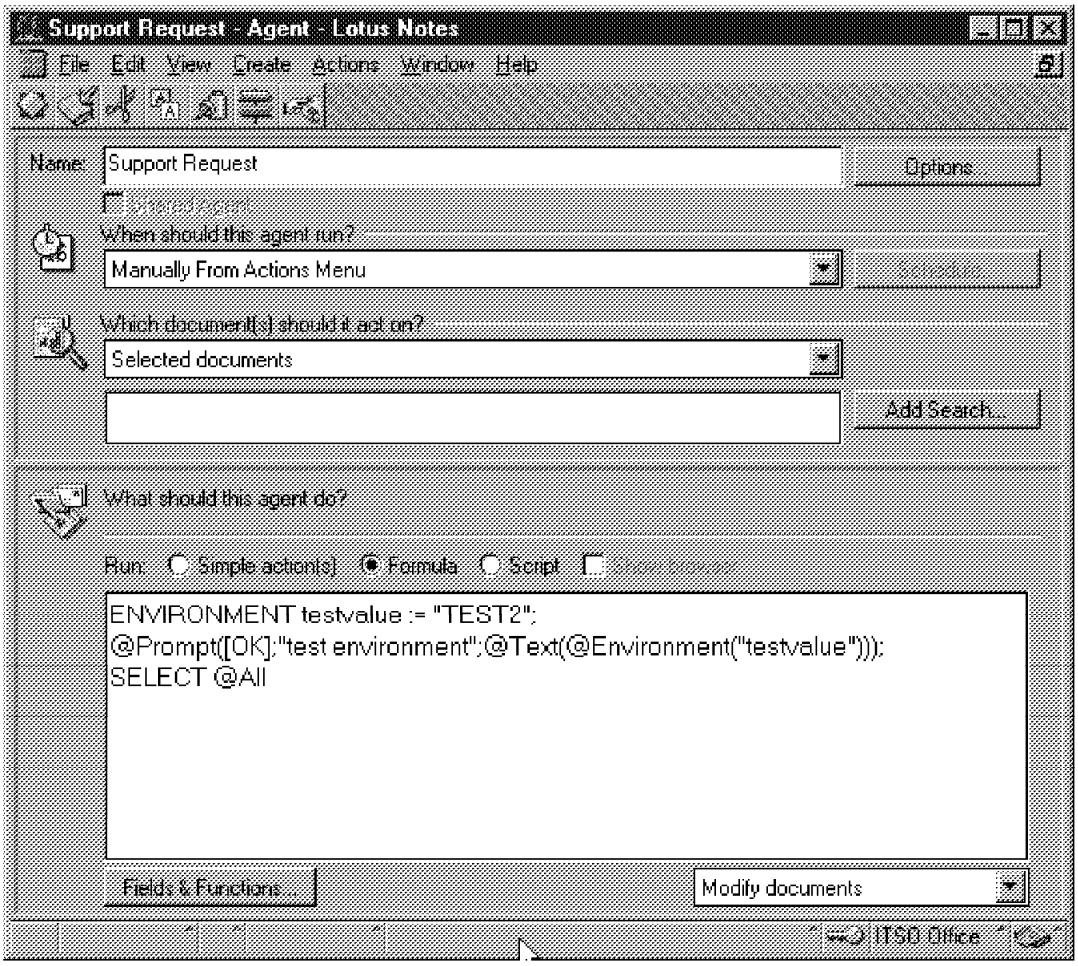


Figure 22. Agent to Change Environment Variable

Executing the agent will prompt an execution security alert. The user can then:

- Abort the action
- Execute the action once
- Change the ECL options to trust the signer to perform this action and execute

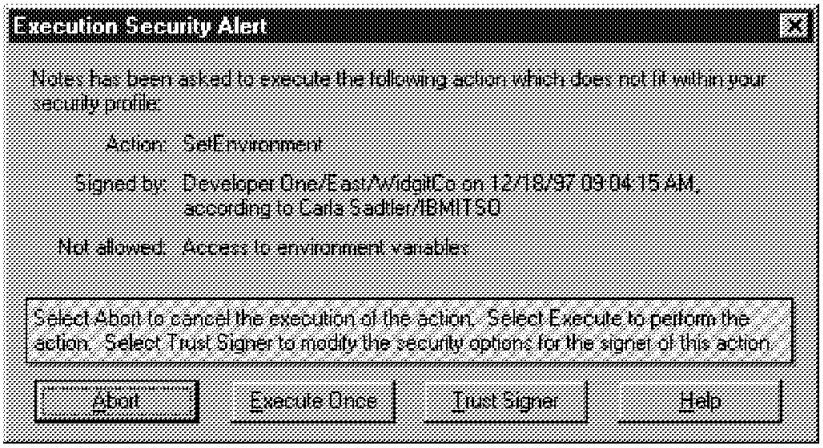


Figure 23. Execution Security Alert

Chapter 4. World Wide Web Security Features

The World Wide Web offers many security challenges compared to Notes. In part this is because of the nature of the application; it is a lightweight, stateless protocol that does not lend itself to strong authentication methods. However, the main reason why the Web gives security concerns is because of the network in which it usually operates, the Internet. The Internet is not a single network, but is comprised of a number of IP networks linked by backbone routers. This results in a network with widespread public access that has no central point of management control. There is a whole community of sad human beings who use this free-and-easy environment as a playground in which to test their system-cracking skills.

What do you call the bad guys?

Finding a suitable label for the people who attack computer systems on the Internet can be a problem. The news media tend to use the term *hacker* but a lot of the technical community dislikes this usage. To them a hacker is the doyen of systems administrators, someone who can dive into the kernel code of a UNIX system and make the machine do things it was never designed for. (At this point you may have a mental picture of someone with long hair and wearing sandals. These are not mandatory hacker attributes, but they do help the image.) Clearly we don't want to confuse intellectuals of this kind with the kind of low-life who breaks into computer systems to steal or vandalize. The term for these people most commonly accepted in Internet security circles is *crackers*.

We divide the techniques for protecting Web traffic into two categories:

Application protection This is equivalent to the Lotus Notes techniques that we described in Chapter 3, "Lotus Notes Security Features" on page 15. It deals with providing access control, authentication, confidentiality and data integrity at the Web application layer. Some of the techniques offer limited protection, such that they may be appropriate in an intranet environment, but are too dangerous to use in the Internet. We highlight these cases as we go.

End-Point Protection No matter how well you protect the application traffic itself, you are always vulnerable to attack in the machines that provide the session endpoint. These are usually either *firewalls* which separate the Internet from your private network or exposed servers.

In this chapter we describe the facilities provided by the Web for application protection. We look at firewall configurations in Chapter 7, "Lotus Notes Firewall Configurations" on page 121. For more detail about Web security questions, refer to *www.security: How to Create a Secure World Wide Web Connection*, SG24-4564.

© Copyright IBM Corp. 1997

4.1 Web Security Facilities

The application level communications protocol used by the World Wide Web is the Hypertext Transfer Protocol (HTTP). HTTP includes a simple user ID and password-based authentication scheme known as *basic authentication*. The implementation of basic authentication is server-specific, but in general they all use it for two purposes:

- As a mechanism to identify which user is accessing the server
- To limit users to accessing specific pages (identified as Universal Resource Locators, URLs)

Basic authentication is an attempt to address two of our security objectives, access control and authentication. However, it does not address questions of confidentiality or data integrity. For these objectives a protocol that uses cryptographic techniques is needed. There are several protocols that seek to meet these needs, but only one is universally implemented, namely the *Secure Sockets Layer* (SSL). We describe the operation of basic authentication and SSL in the following sections.

4.1.1 HTTP Basic Authentication

Basic authentication is based on user IDs and passwords, rather than public key cryptography as in the case of Notes authentication. You have to configure the server to identify which parts of the document tree are protected. These zones of protection are known as *realms*. Each realm is associated with a set of user IDs and passwords that are allowed access. Realms can contain any kind of server object, such as CGI programs, as well as HTML pages.

The detailed mechanism by which realms are defined varies from one server type to another. If you want to see an example of how to configure basic authentication for the IBM Internet Connection family of Web servers, refer to *www.security*, SG24-4564. In the case of the Domino server, the basic authentication realms are defined using normal Notes access controls. We discuss this in detail in Chapter 5, “Security in the Domino Server” on page 51.

4.1.1.1 How Basic Authentication Works

Basic authentication uses a challenge mechanism to prompt users to authenticate themselves. Figure 24 on page 39 illustrates the process.

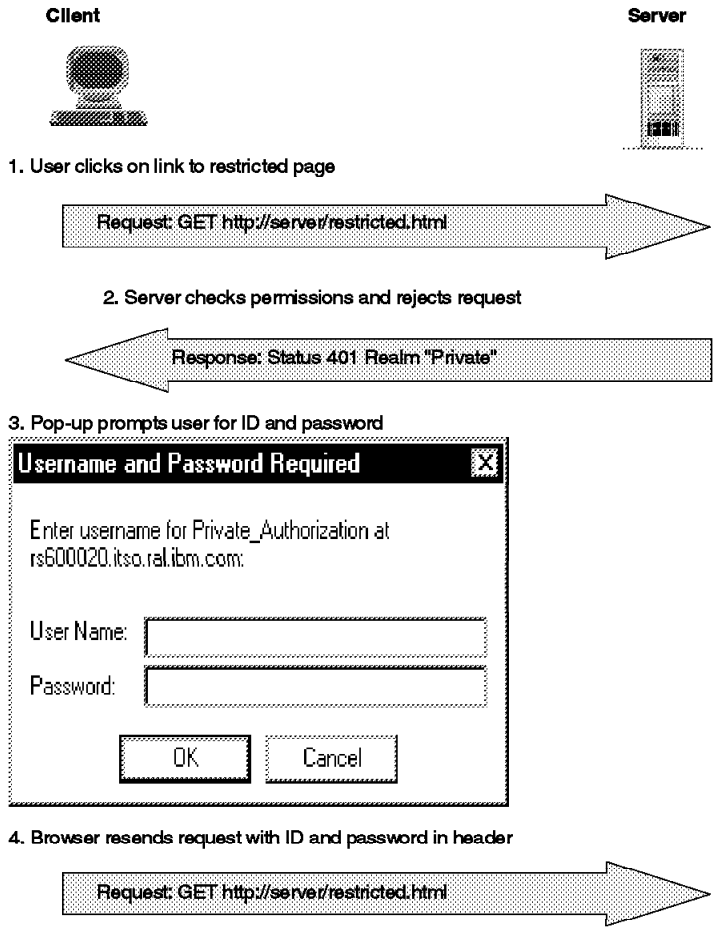


Figure 24. HTTP Basic Authentication

The diagram shows that when a client requests a URL, the server checks to see if the URL requires user authentication. If it does, the server rejects the request, with a status code of 401. The browser then pops up a dialog box on the user′s screen, asking for a user ID and password. When the user has provided them, the browser resends the original request, but with the addition of the following MIME element within the HTTP header:

```
Authorization: Basic <userID and password block>
```

The user ID and password block is constructed by creating a string of the form: userID:password and then encoding it using the base64 algorithm.

You may wonder, given the above description, why you are not repeatedly prompted for a password every time you access a new restricted page. The reason is that the browser caches the user ID, password, server name, and realm name in memory, so that if it receives another 401 status code for the same server/realm combination it can reissue the request using the appropriate user ID and password. In fact, most browsers go one stage further than this and send a user ID and password for any URL that is *likely* to need it. The way this is implemented varies from one browser to another, for example:

- Netscape Navigator sends the information with any URL that is in the same logical directory. For example, imagine that the user requests http://hostX/secret/foo.html and is prompted for a password with realm name PRIVATE. Netscape Navigator will store the details in memory. Later, if the

user clicks on a link to URL http://hostX/secret/bar.html, the browser will send the user ID and password for realm PRIVATE in the request, without waiting for a 401 challenge from the server.

- Lotus Notes Web Navigator uses a similar technique, except that it sends whichever user ID and password it most recently used for the target server.
- Microsoft Internet Explorer behaves in the same way as Netscape.

The objective of these tricks is to reduce network traffic and improve responsiveness, by eliminating a number of invalid requests and 401 status responses. They also, unfortunately, have the side-effect of re-transmitting the user ID and password when it may not in fact be necessary.

4.1.1.2 Is Basic Authentication Secure?

There are two obvious loopholes in HTTP basic authentication:

- The user ID and password are included in the packet header, which means that they can be captured by anyone with a network sniffer or trace tool at any place in the session path.
- The user ID and password are cached in the browser, so if you leave the machine unattended anyone can use your ID to access restricted information.

The second loophole is no different from any other situation where a machine is left unattended. The solution is one of user education: always lock the screen when not at your desk. Note that the caching is in memory, so the user information is lost once the Web browser has been shut down.

The first loophole is more significant. The user ID and password are not encrypted when they are placed in the packet header, but instead are encoded with base64. Base64 is an algorithm that forms part of the Multipurpose Internet Mail Extensions (MIME) protocol. It is a mechanism that turns any bit stream into printable ASCII characters. (It is described in RFC1521.) In fact, the objective of base64 is not for masking data at all, but to provide a method to send binary data through a mail gateway that can only handle character data. The result of this is that by capturing the Authorization: Basic header from an HTTP request, an attacker can easily extract the user ID and password.

How serious is this exposure? Within a corporate network it may not be a big problem. In fact, base64 offers protection of user IDs and passwords that is superior to many older protocols that send them as clear text. In the Internet it is a different story. Here you have to assume that someone, somewhere is tracing everything you send. Clearly HTTP basic authentication should not be used as the sole method of protection for any critical resource.

You can make basic authentication secure by providing an encrypted connection for it to operate in. SSL is a good example of a protocol that encapsulates HTTP data in this way. In Chapter 5, “Security in the Domino Server” on page 51 we see how the Domino server uses HTTP basic authentication, including examples of using SSL to protect the password data. We describe the operation of SSL in the next section.

4.1.2 Secure Sockets Layer (SSL)

The SSL protocol was originally created by Netscape Inc., but now it is implemented in World Wide Web browsers and servers from many vendors. SSL makes use of a number of cryptographic techniques, such as public key and symmetric key encryption, digital signatures and public key certificates. If you want to learn a little more about these things, you may wish to read Appendix A, “Cryptographic Techniques” on page 149 before proceeding.

SSL has two main objectives:

1. To ensure confidentiality, by encrypting the data that a client and server send.
2. To provide authentication of the session partners, using RSA public key methods. Most current implementations only require the server to be authenticated in this way, although the protocol does allow for client authentication. We discuss this in more detail in 4.1.2.5, “SSL Client Authentication” on page 46.

Although SSL is normally used to provide secure encapsulation of HTTP, it can be applied to any TCP/IP application. A number of implementations for other protocols have been developed.

There are two parts to SSL:

- The *handshake*, in which the session partners introduce themselves and negotiate session characteristics.
- The *record protocol*, in which the session data is exchanged in an encrypted form.

4.1.2.1 The SSL Handshake

Figure 25 on page 42 shows a simplified version of the SSL handshake.

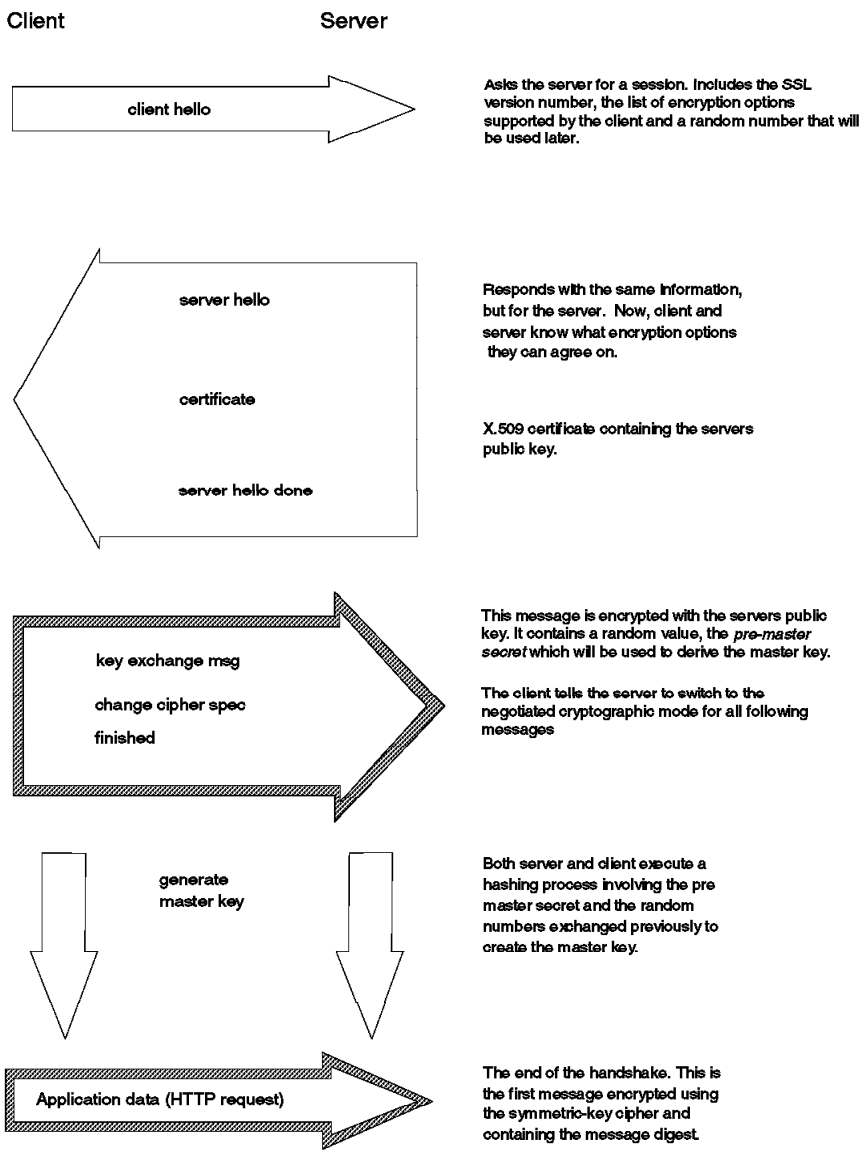


Figure 25. SSL Handshake Process

The two *hello* messages are used to exchange information about the capabilities of the client and server. This includes a list of *ciphersuites*, combinations of cryptographic algorithms and key sizes that the client and server will accept for the session. Also the server provides a public key certificate. This is the method by which SSL checks identity and authenticity of the session partner. In this example we only show the steps for server authentication, but if client authentication was required there would be another message exchange using the client public key. Finally the session partners separately generate an encryption key, the *master key* from which they derive the keys to use in the encrypted session that follows.

You can see from this example that there is significant additional overhead in starting up an SSL session compared with a normal HTTP connection. The protocol avoids some of this overhead by allowing the client and server to retain

session key information and to resume that session without negotiating and authenticating a second time.

4.1.2.2 The SSL Record Protocol

Once the master key has been determined, the client and server can use it to encrypt application data. The SSL record protocol specifies a format for these messages. In general they include a message digest to ensure that they have not been altered and the whole message is encrypted using a symmetric cipher. Usually this uses the RC2 or RC4 algorithm, although DES, triple-DES and IDEA are also supported by the specification.

The U.S. National Security Agency (NSA), a department of the United States federal government imposes restrictions on the size of the encryption key that may be used in software exported outside the U.S. These rules are currently under review, but the present effect is to limit the key to an effective size of 40 bits. The RC2 and RC4 algorithms achieve this by using a key in which all but 40 bits are set to a fixed value. International (export) versions of software products have this hobbled security built into them. SSL caters for mismatches between the export and nonexport versions in the negotiation phase of the handshake. For example, if a U.S. browser tries to connect with SSL to an export server, they will agree on export-strength encryption.

4.1.2.3 Using SSL in Practice

The negotiation and authentication process of the SSL handshake is rather complex, but fortunately it is transparent to the user. In fact, all that a user has to do to enter an SSL connection is to alter the URL prefix from http: to https:. This acts as a trigger to the browser software to start the SSL handshake. Once the SSL connection has been established the browser gives the user a visual indication. In the case of Netscape Navigator this is a key symbol at the lower left of the screen (see Figure 26).

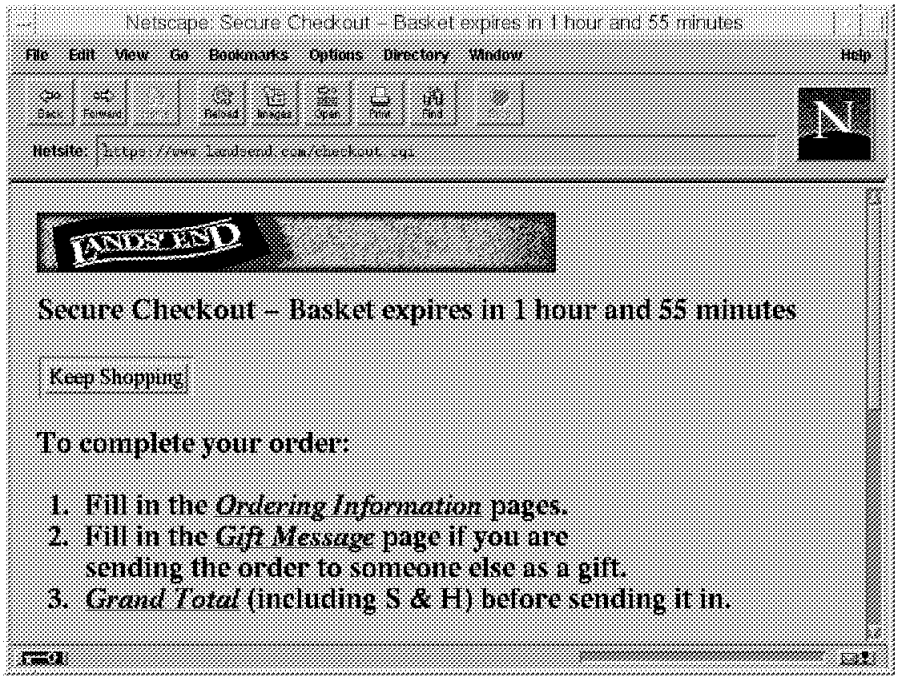


Figure 26. SSL Session Indicator in Netscape

From the point of view of the Webmaster, SSL is also quite simple. First the Webmaster needs to generate a key pair for the server, and obtain a certificate for it. Normally this involves providing documentation to a certifying authority and paying an annual fee, although it is also possible to generate your own certificates for testing and intranet use. We discuss how SSL uses certifying authorities below, and we describe the specific certificate handling capabilities of the Domino server in 5.2.1, “Invoking SSL on Domino” on page 57.

Once the server certificate has been installed, the Webmaster can create HTML links with an https: prefix to cause SSL to be invoked. For example:

```
<A HREF=https://my_server/secret.doc>Go into SSL</A>
```

4.1.2.4 SSL and Certifying Authorities

You can see from Figure 25 on page 42 that authentication in SSL depends on the client being able to trust the server's public key certificate. A certificate links the description of the owner of a key pair to the public part of the key. The validity of a certificate is guaranteed by the fact that it is signed by some trusted third party, the *certifying authority* (CA). But how does a certifying authority become trusted? In the case of an SSL-capable browser, the certificates of trusted authorities are kept in a key database, sometimes called a key ring file. The list of top-level authorities is pre-installed when you get the browser. Figure 27 shows part of the list of CA certificates provided by Netscape Navigator.

Figure 27. Certifying Authorities in Netscape Navigator

This approach has the benefit of being very simple to set up; a browser can authenticate any server that obtains a public key certificate from one of the CAs in the list, without any configuration or communication with the CA required. However, there are also some problems arising from this method. The first is that a new CA will not automatically be recognized until the browser (wherever it may be in the world) has been updated. The second problem is that there is no way for certificate revocations to be processed. (For example, if a CA

determines that a public key owner is fraudulent after a certificate is issued, the certificate will remain useable until it expires, without the end user being aware of any concern.)

The browser vendors have a two-part scheme to overcome the first problem (new CAs):

1. There is a special MIME format, application/x-x509-ca-cert, which allows a browser to receive a new CA certificate that has been signed by one of the known CAs. However, this method is not widely implemented yet. (See http://home.netscape.com/eng/security/downloadcert.html for details of download formats.)
2. The browsers will tell you that you are connecting to a secure server whose certificate is not from a known CA. You can then elect to trust just that server (that is *not* the CA that signed the server′s certificate).

This latter approach is useful in an intranet environment, where you may want to create an enterprise-only CA. For Internet applications you should purchase a certificate from one of the known CAs, such as VeriSign. Figure 28 and Figure 29 on page 46 show how Netscape Navigator warns you about a site that has a certificate from an unknown CA, and then allows you to choose to trust the site anyway.

Figure 28. Server Certificate Signed by Unknown CA

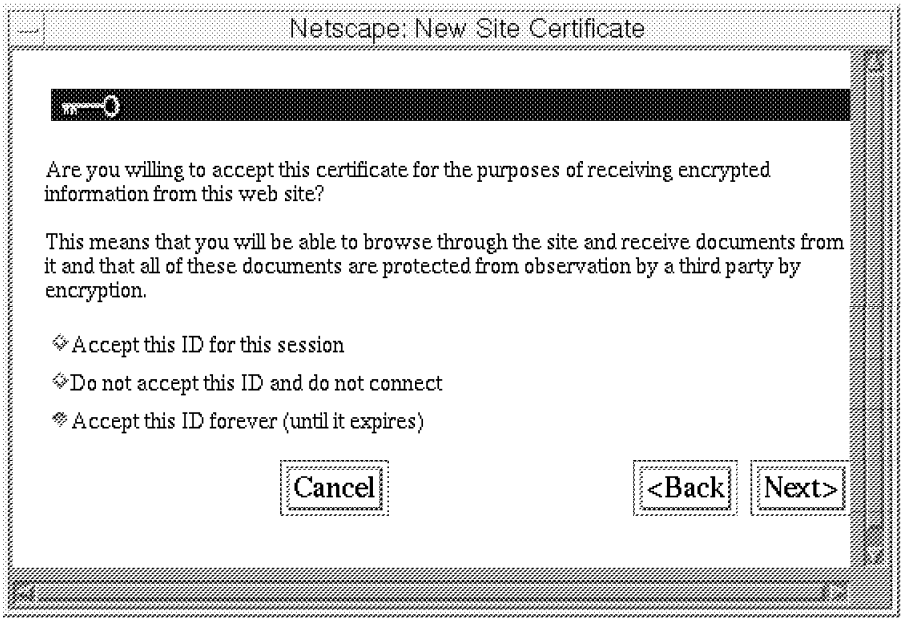


Figure 29. Option to Add Trusted Server. Note that this will not make the CA trusted, so you will see the same warnings if you access a second server with a certificate signed by the same CA.

4.1.2.5 SSL Client Authentication

In the release 4.5, Domino does not make use of SSL client authentication; instead it uses SSL to provide an encrypted channel through which HTTP basic authentication user IDs and passwords can be exchanged safely. However, this is a rapidly-developing area and we expect that support for SSL client authentication will be added to Domino R4.6. Therefore we describe the way that client authentication works and the product developments that will support it.

In 3.2.5, “Authentication and Validation” on page 26 we showed that there is symmetry in the way that Notes performs authentication. Both client and server use digital signatures to identify themselves and those signatures use public keys that are validated by the existence of a shared hierarchy of certificate authorities. You may think that SSL could apply a similar approach to allow Web browsers to authenticate themselves, simply by having the client implement a mirror image of the server authentication process. In fact, this is exactly what the SSL protocol specifies, by adding to the handshake a server challenge that the client must encrypt using its private key.

Unfortunately life is not that simple. Notes can support a symmetrical authentication scheme because you have control over the CA hierarchy and therefore you can ensure a region of shared trust. On the Web you do not have that luxury. The monolithic certification scheme used by servers (discussed in 4.1.2.4, “SSL and Certifying Authorities” on page 44) is not flexible enough to support the large number of certificates that client authentication would require.

To enable this there are two problems to be solved:

1. How can you serve a certificate to a browser without too much overhead and without requiring the user to go through a complex registration process?

2. How can you make the process scalable, so that you are not dependent on a small number of top-level CAs?

A consensus has emerged on how these questions will be answered although, at the time of writing, products to implement the answers are still under development.

Serving Certificates to Browsers: A public key certificate provides proof of identity and it is a reasonable assumption that the level of proof needed for a client is much lower than that needed for a server. Before providing a server certificate, the CA will require documentary proof of the legitimacy of the request. In the client case, this proof can often be provided online, because a lower level of checking is needed. This is especially true of an intranet environment and certificate server products are initially intended for organizations that want to set up an internal authentication process.

Netscape has developed a protocol for serving client certificates that is supported by current versions of Navigator. Figure 30 on page 48 shows how it works. The key to the mechanism is the <KEYGEN> tag, an HTML extension that only applies within a form. When the browser sees this tag, it generates a key pair and returns a certificate request for the key pair with the form when it is submitted.

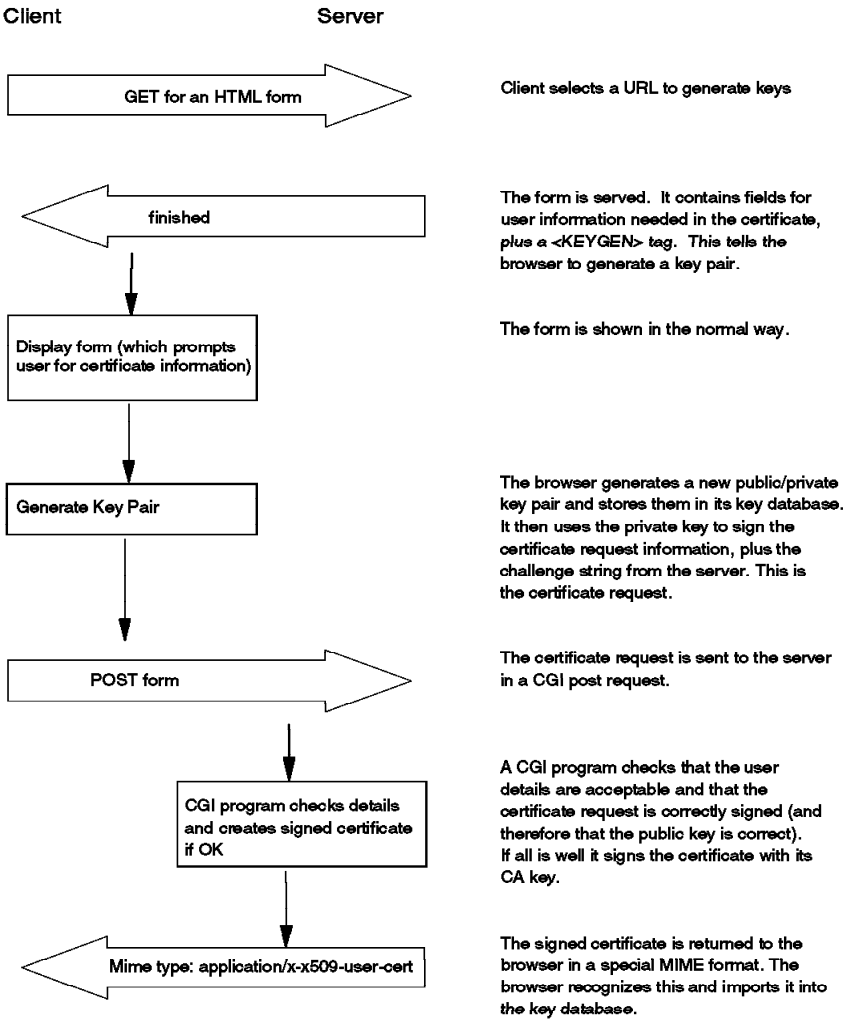


Figure 30. Netscape Method for Installing a Client Certificate

Making a Scalable Certificate Authority Scheme: SSL authentication relies on the session partners recognizing the CA that signed a certificate as a trusted entity. As we described in 4.1.2.4, "SSL and Certifying Authorities" on page 44, there are limitations in the way that this is currently implemented. The challenge is to create a scheme in which CAs can exchange information with other CAs, clients and server systems. This information includes information about new CAs and revoked certificates. In fact this is not just a requirement for CA function, but a more general requirement for the exchange of all sorts of directory information.

There is already an OSI standard for exchange of directory information, the Directory Access Protocol (DAP) which is part of the X.500 standard. However, DAP is exclusively for X.500 directory servers and it makes demands on hosts and networks that limit its applicability, for example:

- DAP requires a full OSI communications and authentication architecture.
- It uses the full X.500 data model, which is more complex than required by most applications and which carries a heavy penalty in code complexity.

The net result of these demands is to prohibit the use of full X.500 DAP implementations on small desktop operating systems. To overcome this, a

subset of DAP, the Lightweight Directory Access Protocol (LDAP) has been developed. LDAP is specified in RFC1777. LDAP operates over TCP/IP, discards the requirement for OSI security, and employs simple string-based data encoding. This means that LDAP will be accessible from systems of all kinds, including regular Web clients.

Lotus has announced that support for LDAP will be incorporated in future Notes products, meaning that directory information from the name and address book can be shared. At that point it will be possible to integrate SSL client authentication with the existing authentication methods of Notes.

4.2 Comparisons Between Notes Security and SSL

In the preceding section and in Chapter 3, “Lotus Notes Security Features” on page 15 we have covered a lot of ground. Table 4 attempts to summarize some of this, comparing security functions at the application level in the Notes and World Wide Web environments.

Table 4 (Page 1 of 2). Comparison of Security Functions in Notes and the Web

Function	Notes Implementation	WWW Implementation
Access control	Multi-level access control based on user ID. Full control over what the client can see and do at the field, view, form and database levels.	Access control to specific documents and executables based on user ID. Single level of control (view or not, execute or not). Access control methods based on SSL client authentication not yet developed.
User management	Distributed directory of user information (name and address book) includes personal details and public-key certificate information. Ability to group users using distributed group profiles.	1. Basic authentication: user ID and password file maintained separately on each server. Ability to create simple groups of users. 2. SSL client authentication: directory of user information will include personal details and public-key certificate information. LDAP will allow cross-directory access.
Certification scheme	Public-key certification authority function provided by domain and subdomain servers. Integrated into the Notes directory. Cross-certification allows authentication across multiple domains, in a controlled fashion. Manual registration by domain administrator.	1. Current (server only) SSL certification: Small number of well-known certification authorities allow servers to be recognized worldwide. 2. Future SSL certification. Certificate servers, linked to X.500 directory servers will allow self-registration by acceptable clients. LDAP will provide distributed access to directory and certificate information.
Authentication	RSA public key technology used to digitally sign a challenge string.	RSA public key technology used to digitally sign a challenge string.

Table 4 (Page 2 of 2). Comparison of Security Functions in Notes and the Web

Function	Notes Implementation	WWW Implementation
Encryption methods	RSA public key technology used to encrypt key material which is subsequently used in a symmetric cipher for bulk data encryption. Encryption integrated with access controls, so that only sensitive information is encrypted. 64-bit keys used for encryption, with 24 bits accessible by the US intelligence agencies in non-US versions of Notes.	SSL: RSA public key technology used to encrypt key material that is subsequently used for bulk data encryption. SSL may be invoked on specific HTML links, but in typical implementations it is not used selectively. 128-bit key usually used for US version, 40-bit in non-US version.

Chapter 5. Security in the Domino Server

The Domino server allows you to use a Web browser in place of a normal Notes client, by mapping between Notes document formats and HTML. As each environment has different security facilities you would not expect an exact match between them. In this chapter we look at the ways in which the two security schemes interact.

You may refer to Chapter 4, “World Wide Web Security Features” on page 37 for further details of Web security.

5.1 Access Control and Authentication

There are two types of Domino Web users, registered users and unregistered users.

A registered user has a person document in the Public Address Book with his/her name and HTTP password. Domino refers to this document and password to authenticate a Web user. An unregistered user is given the name Anonymous ACL in the same no password.

To control the access of Web users, you may specify the Web username or Anonymous in an manner as you would for a Notes user.

5.1.1 How and When Authentication is Performed

Authentication of Web users is done by a technique known as basic authentication. For example, when Web users try to open a database that has a default access of No Access, they are asked by the Domino server to supply a valid name and password (Figure 31). Authentication succeeds if the username and password supplied match the appropriate fields in the Person document. The user is recognized as a registered Web user.

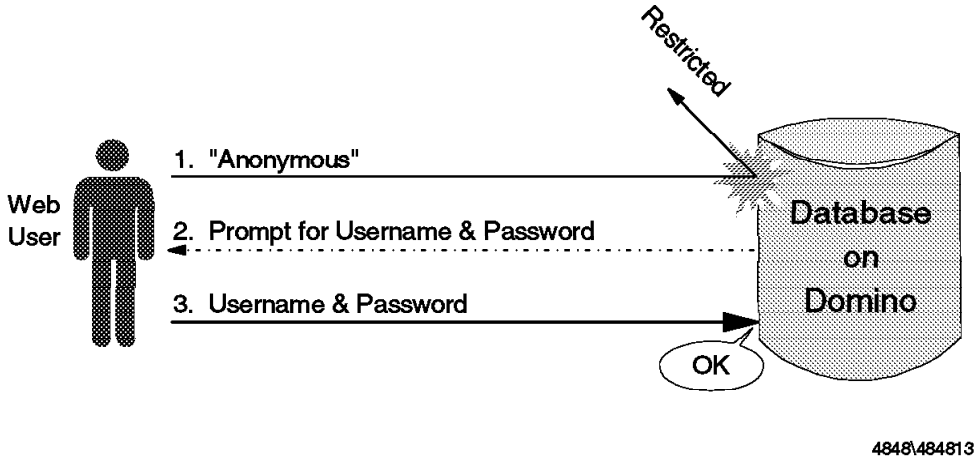


Figure 31. Basic Authentication with Domino

Web users are only authenticated when they attempt to do something for which access is restricted. In other words, if the attempt is not restricted, Web users

© Copyright IBM Corp. 1997

are not required to supply the username nor password and are recognized as Anonymous, even if they are registered.

If there is an entry for user Anonymous in the target database's ACL, it defines an unregistered user's access level. If there is no entry for Anonymous, such users receive the default access.

For example, assume that a Web user who is registered in the Public Address Book, tries to open a database where the default access is No Access and user Anonymous is assigned reader access. He or she can open the database, views, and browse documents without entering a username and password because as Anonymous he or she has reader access, which is enough to perform these operations. When he or she tries to create a new document Domino will ask him or her to supply the username and password because author access or higher is required to create new document.

Note: After a username and password is entered the Web browser remembers the information and re-sends it automatically in response to Domino's requests. In effect the Web session holds the username and appropriate access throughout the session.

After authentication is done, the user's session holds the username as the CGI environment variable REMOTE_USER. Domino interprets the REMOTE_USER variable as a username within the Notes world, and Notes applications receive the interpreted username. For example, the @Username function receives the Web user's name from REMOTE_USER.

If the REMOTE_USER value is undefined (or null), in other words a username has not yet been entered, Domino gives the special name Anonymous to the user session and Notes applications recognize the user as Anonymous.

Note: Notes applications designed for Domino may handle the CGI REMOTE_USER variable directly. Notes applications that are designed for both Notes users and Domino users can use @Username to receive the Web user's name. An example of CGI variables in a Notes application can be seen in 2.2.3, "CGI Variables" on page 13.

The following figure shows the differences of authentication among three types users: Notes users, registered Web users, and unregistered Anonymous Web users.

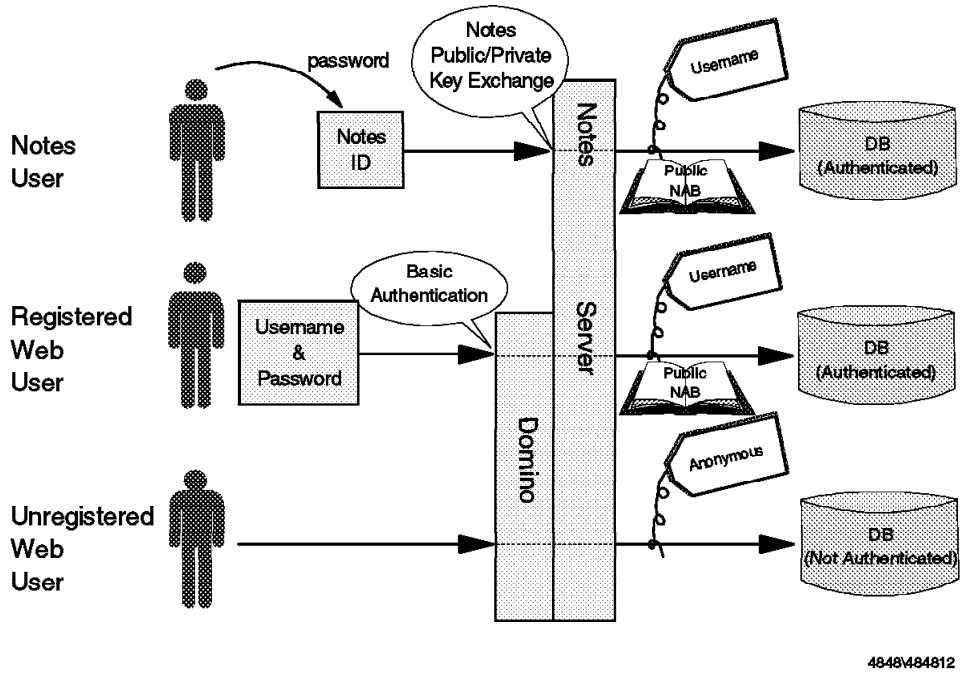


Figure 32. Notes User and Two Types of Web User. Notes users accessing Notes servers enter the password associated with their user.id file. Registered Web users enter a name and http password found in a Person document in the Public Address Book. Web users who are not registered or who have not logged in are treated as though they have the user ID Anonymous. No password is associated with the Anonymous ID.

5.1.2 Defining Web Users

You may register new Web users from scratch, or let existing Notes users have access to the Domino Web server.

To register the user from scratch, follow these steps:

1. Create a new Person document.
2. Enter the required information. The User name and HTTP password fields are the only fields used for Domino Web user authentication. The other fields are optional with the exception of the Last name field. Even though this field is not used in Domino authentication the form won't be saved without it.
3. Save and close the document.

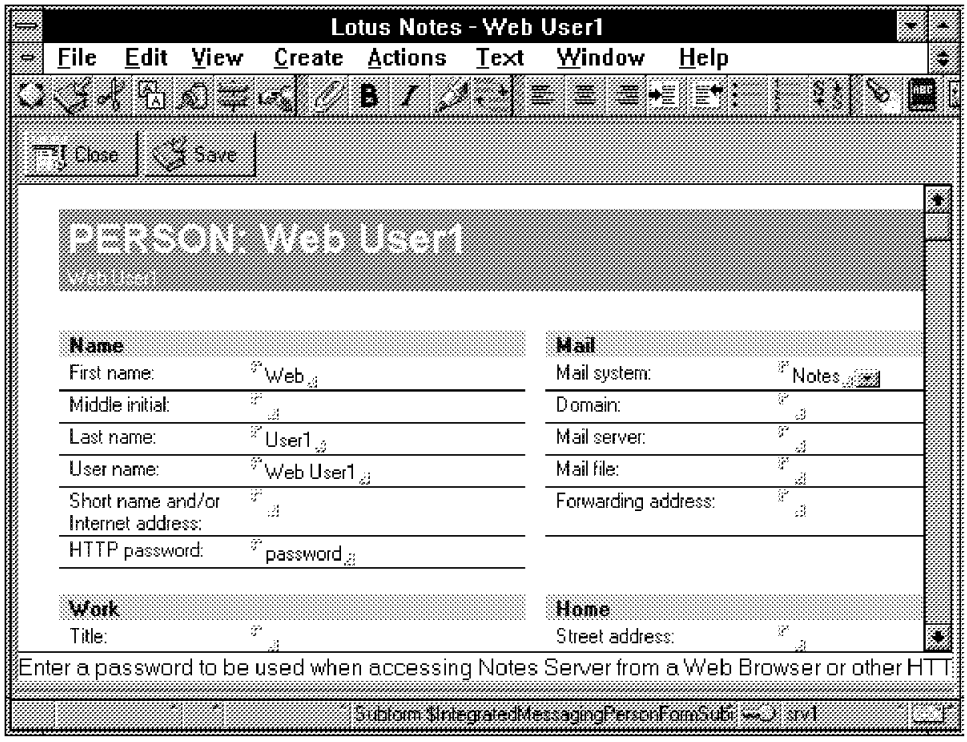


Figure 33. Creating New Web User

To give an existing Notes user Web access:

1. Open the **Person** document of the target user in edit mode.
2. Put a password into the HTTP Password field.
3. Save and close the document.

Attention: Make sure you are aware of the implications of giving Notes users a Web password. You may be inadvertently increasing your security exposure. The authentication offered by the HTTP user ID and password is not as secure as that of the Notes ID. For example, Notes users must have physical access to their notes.id file but a Web user's password may be stolen particularly if it is sent unencrypted over the network.

The password placed in the HTTP password field of the Person document is one-way encrypted (or *hashed*) before saving. Only the hashed form is stored, so you cannot extract a user's password if it is forgotten, only replace it.

5.1.3 Web User Capabilities

Web users manipulate Domino databases using a *Domino URL*. Domino doesn't support all operations available to a Notes client. For example, you can open a view or document the same as with Notes, but you cannot open a properties infobox. A brief overview of Domino URLs is covered in 2.2.1, "Domino URLs" on page 10.

5.1.3.1 Encryption and Signature

SSL security does not have the capability to encrypt or sign individual parts of a document. It is an all-or-nothing approach, so if SSL is invoked, the whole of a document is encrypted, including any artwork or other media files that may be embedded. For this reason, the Notes capability of encrypting fields in a document does not work with Domino. Similarly there is no ability to digitally sign a section of a form or of a mail message.

5.1.4 Access Control Limitations

Most security features available for pure Lotus Notes are also available for Domino. However, from the viewpoint of access control, there are several significant differences.

5.1.4.1 Server Access List

Server access lists for Notes servers provide security, however, server access lists do not work on Domino for Web users. You can control access on each database, but existence of a database on a server cannot be hidden completely from users with access to the server.

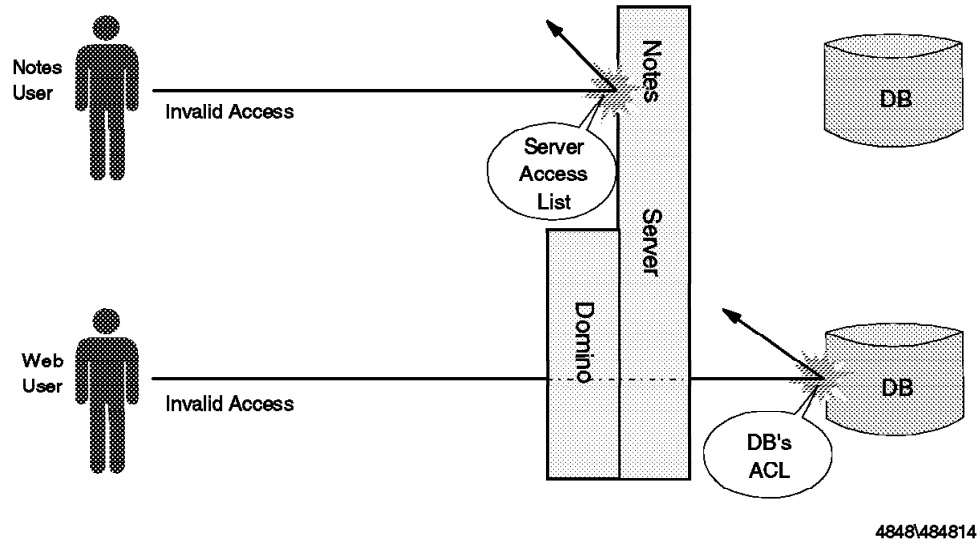


Figure 34. Server Access List Doesn't Work with Domino

Domino servers can be protected from Anonymous user access by setting the field Allow Anonymous Notes connections to No (see 5.3.1.1, "Prohibiting Anonymous Web User Access to Domino Servers" on page 86). If this field is set to Yes, every user (including unregistered Anonymous users) can access the Domino server and *touch* all databases. By specifying the (known or guessed) filename of the database with ?OpenDatabase Domino URL, you can determine whether it exists as a database. For example, if the attempt http://domino.lotus.com/names.nsf?OpenDatabase is restricted to authorized users, the user is asked to enter a valid username and password by Domino. If the specified filename is not a valid database name, the user will receive an error message.

In addition, a user may try to see the database list using /?Open. For example, you can browse the database list of the Domino official site by entering http://domino.lotus.com/?Open. You can have the database name not be shown

in the list by using the database properties infobox (see 5.3.1.5, “Hiding a Database Name from the Database List” on page 91). You can also inhibit database browsing by configuring the server document (see 5.3.1.4, “Inhibiting Web Users from Browsing the Database List” on page 89).

More on the subject of protecting your Domino server can be found in 5.3.1, “Controlling Access to Notes Resources” on page 86.

5.1.4.2 ACL of Directory Link

Domino does not support ACLs in databases reached through directory links. Directory links do work in Domino so it is necessary to closely review databases accessed via directory links (.DIR files). Anything stored in this manner should be open to the public.

5.2 Confidentiality

To implement confidentiality between a Domino server and a Web browser, you must use Secure Sockets Layer (SSL). Without SSL, all data packets are not encrypted and their content can be stolen. We have described how SSL works in 4.1.2, “Secure Sockets Layer (SSL)” on page 41. Figure 35 illustrates the elements that make up an SSL connection.

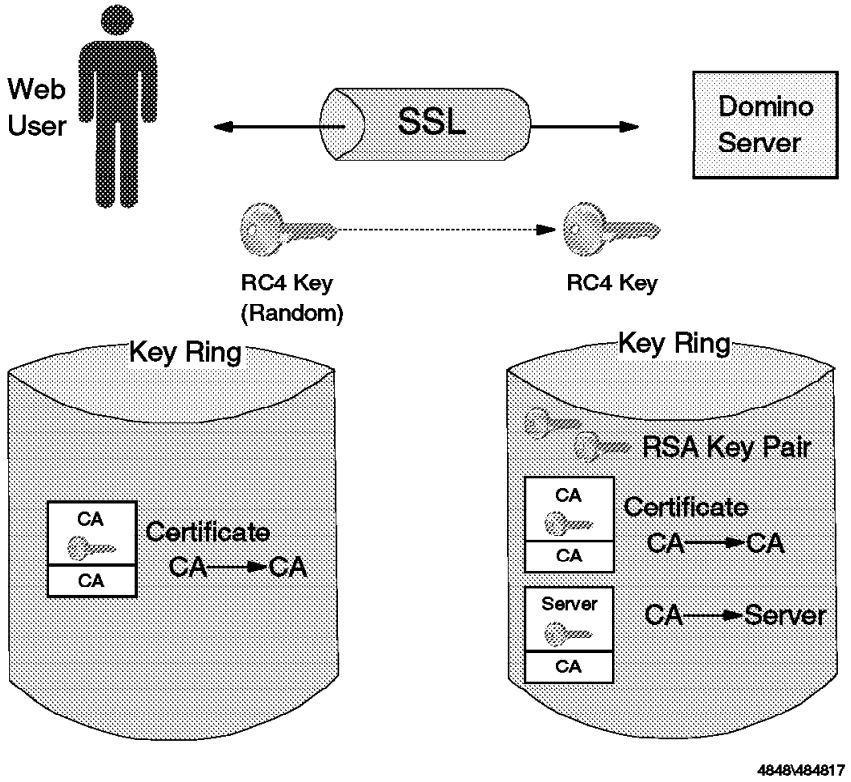


Figure 35. Elements of SSL

Using SSL, all transactions are encrypted by the sender with a symmetric encryption algorithm, such as RC4 or DES, and are decrypted by the receiver. Before communication starts, the symmetric key is transferred from one to another during a handshake phase, using a public key encryption algorithm, such as RSA. Both directions of communication are encrypted with SSL, but

only the server is authenticated using digital signature techniques. This authentication relies on the server having a public key certificate signed by a certifying authority (CA) that the browser recognizes.

5.2.1 Invoking SSL on Domino

To invoke SSL on a Domino server, you must:

1. Prepare a key ring, which contains a public/private key pair and certificates
2. Configure the server document to enable the SSL port

There are several ways to obtain a key ring:

- Use an existing key ring. This is what we did, using a key ring that was provided to IBM for test purposes. However, in general it is not good practice to move private keys between different servers.
- Generate a key ring using the Domino SSL Administration database and:
  - Create self-certified certification (suitable for testing purposes).
  - Send a certificate request to a commercial CA, such as VeriSign.
  - Send a certificate request to an internal CA; Domino itself may act as an internal root CA if you configure it so.

The Domino SSL Administration Database is shipped with the Domino server. You can perform management tasks through this database, such as creating a key pair and a certification request, sending a certification request to a CA or letting Domino certify it, and storing a certificate into a key ring file.

In the following sections, we describe:

- How to install an existing key ring file on Domino
- How to configure a server document in the Public Address Book
- An overview of using the Domino SSL Administration database

5.2.1.1 Installing a Key Ring File and a Password File

We installed an existing key ring file on our Domino server. Our key ring contains a public/private key pair and a certificate. The key pair is for RSA cryptosystem (512-bit key length) and created by IBM for testing purposes. This key ring file has the extension .KYR.

A key ring file is password protected, very similar to the password protection that the Notes user.id file has (as described in Chapter 3, “Lotus Notes Security Features” on page 15). However, in the case of a server it is often not acceptable to require a password to be entered each time the system is restarted, for example, after a power failure. The circumvention for this is to stash a masked copy of the password in another file for the server to use when it starts up. In the case of Domino, the password is stored in a file with a .STH extension, whose basename is the same as .KYR file. For example, the password of KEYFILE.KYR is stored in KEYFILE.STH. Note that this mechanism is different from other Web server software such as the IBM Internet Connection Secure Server.

In our case we knew the password of our key ring file, but did not have the corresponding .STH file. We used a cheap trick to create .STH file. The procedure was as follows:

1. Determine the password and filename of the key ring. Our key ring file is named V1S512.KYR. The password file must be V1S512.STH.
2. Open the **Domino SSL Administration** database and select **Create Key Ring - Self-Certification**. Click on the **Create self-certified key ring** button.

   Note: We just want to create a “dummy” key ring file. There are several ways to create a key ring file in this database. This is the shortest way to do it.

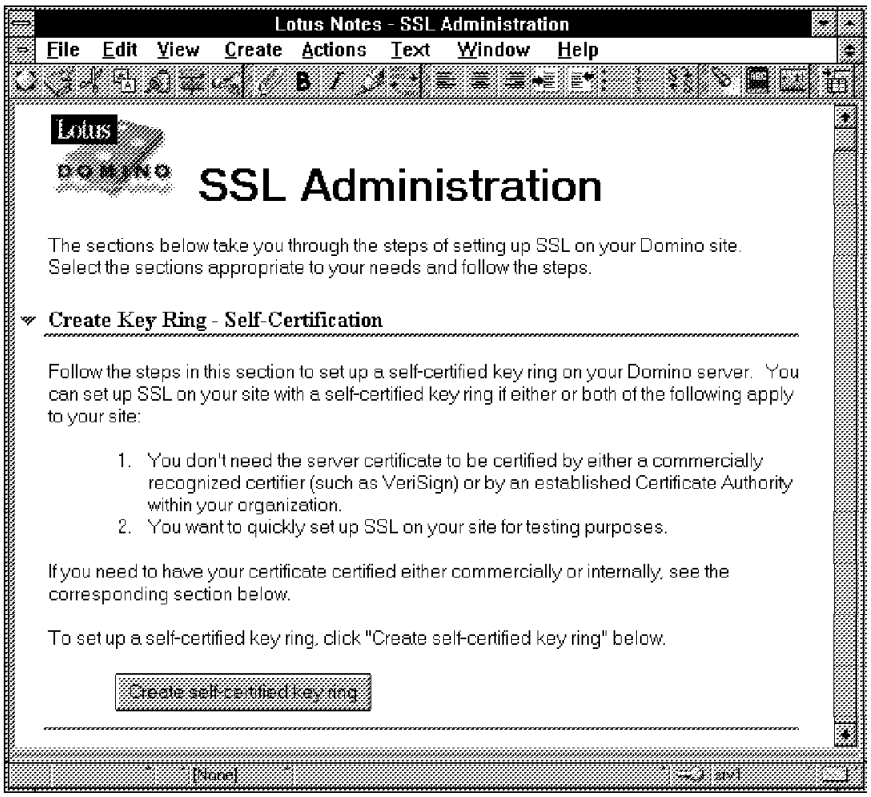


Figure 36. Creating a Dummy Key Ring

3. Fill in the information required. The key ring password is essential. Enter the password for the real key ring (in our case, the password for V1S512).

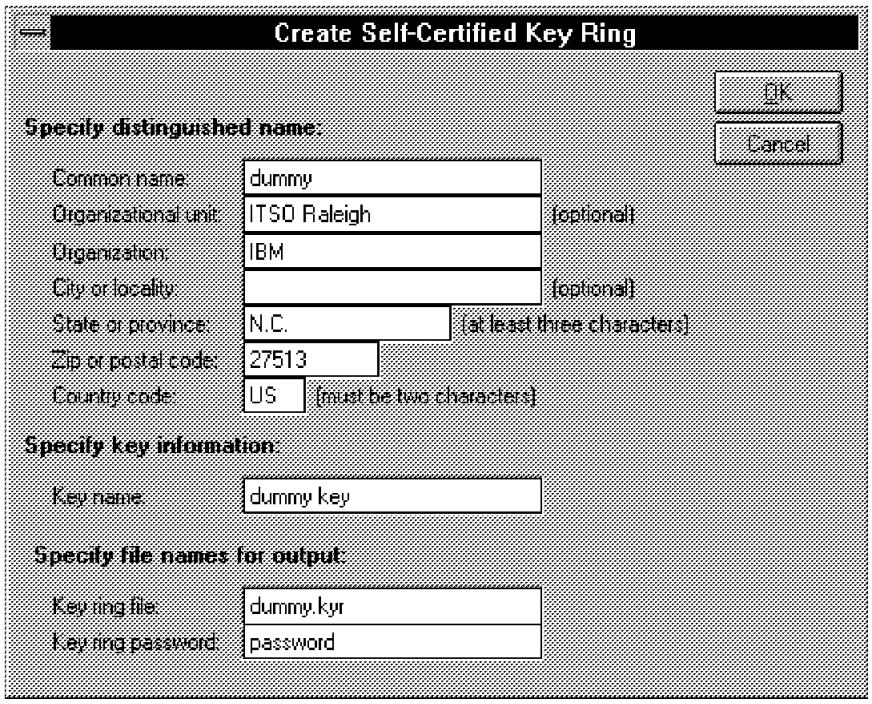


Figure 37. Entering Key Ring Password

4. Click **OK** to proceed. This will create two files, DUMMY.KYR and DUMMY.STH, in the Notes data directory.
5. Now DUMMY.STH contains the password for V1S512.KYR key ring file. Rename DUMMY.STH to V1S512.STH and erase DUMMY.KYR.

5.2.1.2 Configuring the Server Document

In the server document, there are four items to be configured to use SSL (see Figure 38):

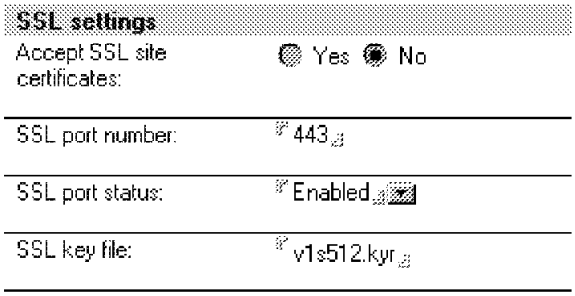


Figure 38. HTTP Server Section of Server Document

Accept SSL site certificates This controls whether Web Navigator should allow SSL connections to sites that do not have certificates signed by a recognized certifying authority. (See 4.1.2.4, “SSL and Certifying Authorities” on page 44 for an explanation of this.)

SSL port number The standard HTTP port number is 80. The standard secure HTTP port is 443 (Figure 39 on page 60). You should use 443 with SSL so you should leave this field as the default value.

SSL port status Switch this field to Enabled to activate SSL on your Domino server.

SSL key file Specify the file name of your server's key ring. Normally the key ring is placed in the Notes data directory. If the location of the key ring (both .KYR and .STH) is not there, you must specify the full pathname of the key ring file, such as C:\DIRECTORY\KEYFILE.KYR.

The result of making these changes is that the Domino server will listen on TCP port 443 as well as the standard TCP 80 port that HTTP (normal Web access) uses. These port details are hidden from the browser user. The session enters SSL mode when the user invokes a URL with an https:// prefix in place of http://. This triggers the browser to start the SSL handshake using port 443 (see Figure 39).

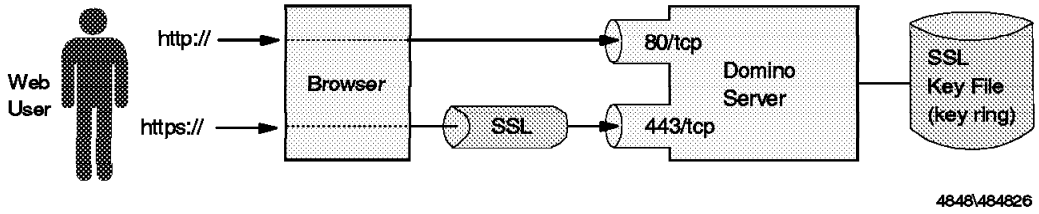


Figure 39. SSL Port Number

5.2.1.3 Using the Domino SSL Administration Database

To illustrate the ability of the SSL Administration database, we discuss two scenarios: one using a commercial CA, and another running Domino as an internal CA.

There are five tasks in this database as shown in Figure 40 on page 61.

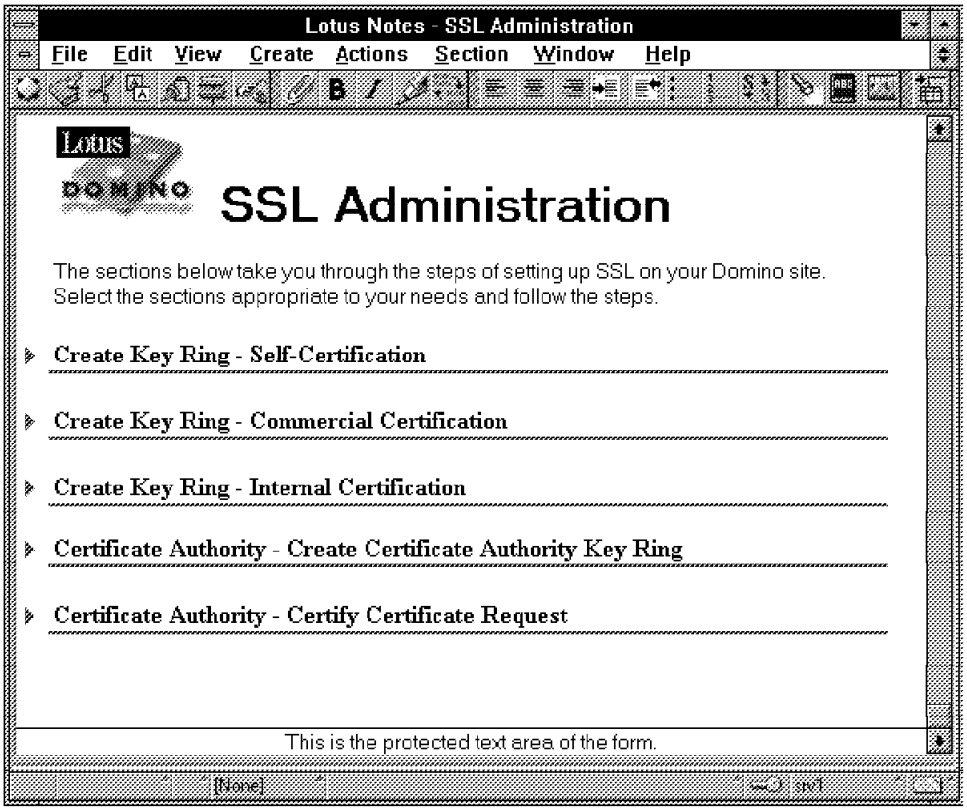


Figure 40. Five Tasks in SSL Administration Database

Create Key Ring - Self Certification This is suitable for the purpose of testing server authentication. You can create a self-certified key ring using this menu. This is an adequate option if you don't need to work with any commercial or internal CAs.

Create Key Ring - Commercial Certification You can create a certification request for VeriSign, send it via e-mail, and read the received certificate into your key ring.

Create Key Ring - Internal Certification This menu item is very similar to the previous one. You can do the same process for internal CAs or commercial CAs other than VeriSign.

Certificate Authority - Create Certificate Authority Key Ring When you want to establish an internal root CA, you may choose this menu. You can create a CA key ring and a CA certification. Both are used to certify certificate requests.

Certificate Authority - Certify Certificate Request When you receive a certificate request, you can certify it with your CA key ring.

Scenario 1: Working with Commercial CAs:

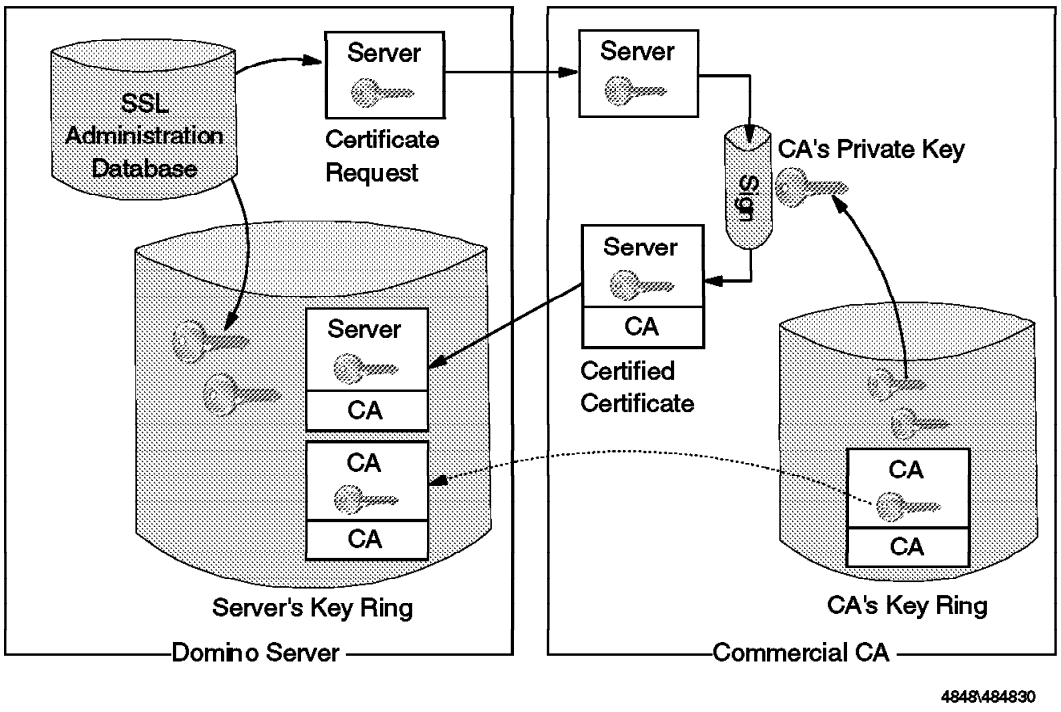


Figure 41. Overview of Scenario 1

This following section describes how to create a key ring file and get valid certification from commercial CA:

1. Open the **SSL Administration** database. This application should be accessed as *local* at the server.
2. Go to the section Create Key Ring - Commercial Certification to choose **VeriSign** as your CA. In this database of the current version, Commercial Certification means VeriSign.

   Note: If you want to work with another commercial CA other than VeriSign, go to Create Key Ring - Internal Certification. The operation is almost the same. Where they differ we add a note.

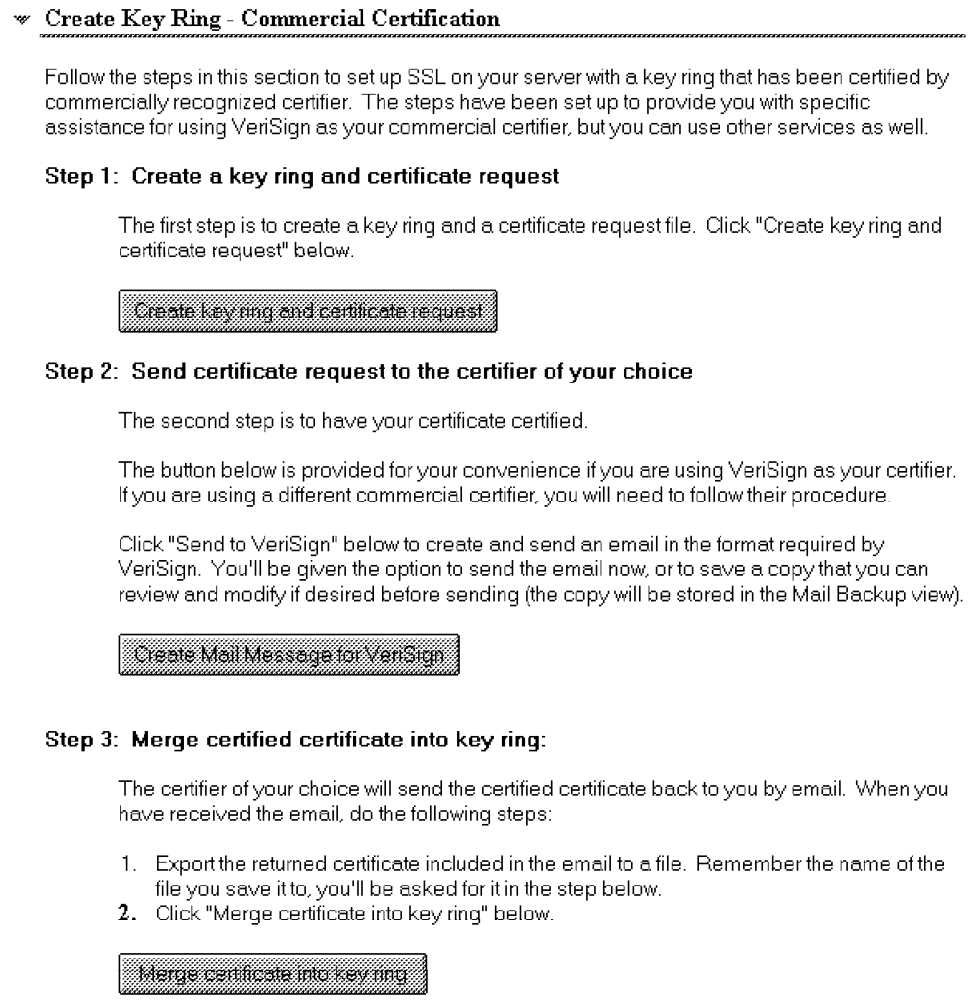


Figure 42. Create Key Ring - Commercial Certification

3. Go into Step 1: Create a key ring and certificate request to create a new key pair and a certificate request.

   Click the button labeled **Create key ring and certificate request**. Enter the required information: common name, organization, country code, and so on (Figure 43 on page 64).

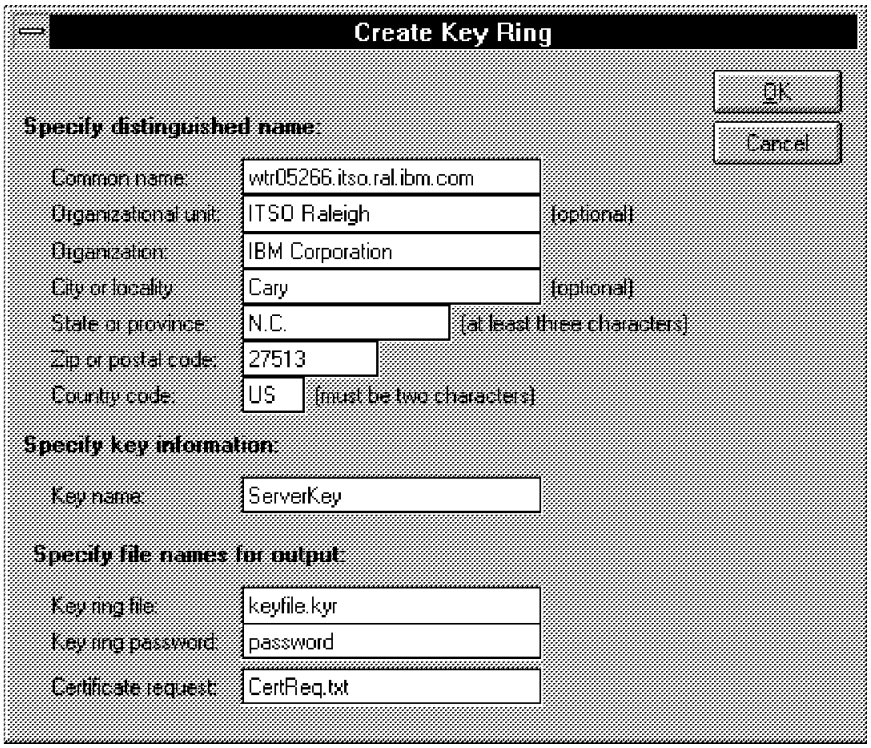


Figure 43. Create Key Ring Dialog Box

Click on **OK** to proceed. The key ring file and certificate request file are created in the Notes data directory, where the database files (.NSFs) reside. An example certificate request file is shown in Figure 44.

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 4,MIC-ONLY
Content-Domain: RFC822
Originator-Certificate:
 MIICCzCCAbUCBDJOhrAwDQYJKoZIhvcNAQECBQAwgZAxCzAJBgNVBAYTAlVTMQOw
 CwYDVQQIEwROLkMuMQOwCwYDVQQHEwRDYXJ5MRgwFgYDVQQKEw9JQkOgQ29ycG9y
 YXRpb24xFTATBgNVBAsTDElUUO8gUmFsZWlnaDEiMCAGA1UEAxMZd3RyMDUyNjYu
 aXRzby5yYWwuaWJtLmNvbTEOMAwGA1UEERMFMjc1MTMwGhcLOTYxMDI4MTAxMFoX
 Czk3MTAyODEwMTBaMIGQMQswCQYDVQQGEwJVUzENMAsGA1UECBMETi5DLjENMAsG
 A1UEBxMEQ2FyeTEYMBYGA1UEChMPSUJNIENvcnBvcmFOaW9uMRUwEwYDVQQLEwxJ
 VFNPIFJhbGVpZ2gxIjAgBgNVBAMTGXdOcjA1MjY2LmlOc28ucmFsLmlibS5jb2Ox
 DjAMBgNVBBETBTI3NTEzMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAM+39H7KvYme
 BHUAU/iYXxgaTcSpd1bzXTwwdVfsUBv2GOUhpxbQcB5Cme+ppVxcrI/IUIfHhABU
 KbQxhXyClCsCAwEAATANBgkqhkiG9wOBAQIFAANBAAqAuoEuukQir4BzMqhH6V+E
 nblkQXTUISxRlEkmng/qvx89U6BQcjT2EObyI4LAAPsbup2yrk1zb7EeO+oUHos=
MIC-Info: RSA-MD5,RSA,
 qFPQ8xS/9pbS6/InSK9SHI12lun8762y+F8qh7g7TblnM5LHypb8J1QwKWYz5l5W
 xjxuupP4oKGbLBTh92C9fg==

VGhpcyBpcyBhbiBSRkMtMTQyNCBDU1IuCg==
-----END PRIVACY-ENHANCED MESSAGE-----
```

Figure 44. Certification Request File Example

4. Go into Step 2: Send certificate request to the certifier of your choice to send a certificate request to a CA.

   Note: If e-mail is not available for requesting and receiving certificates, or if your Notes system has no access to the Internet e-mail, you may handle the certification request file manually instead of taking this step.

   Click on **Create Mail Message for VeriSign** to compose the message for VeriSign.

Enter the required information. All information may be filled in automatically, except for Webmaster information.

Before selecting **OK**, make sure the destination address in the To: field is valid. Note that this message is to be sent via Notes. You may modify this field as needed; for example, add @ INTERNET-GW to specify the gateway to Internet.

Check the Save copy of mail message in Mail view box to compose the message as a draft or to create a back-up copy of the outgoing message.

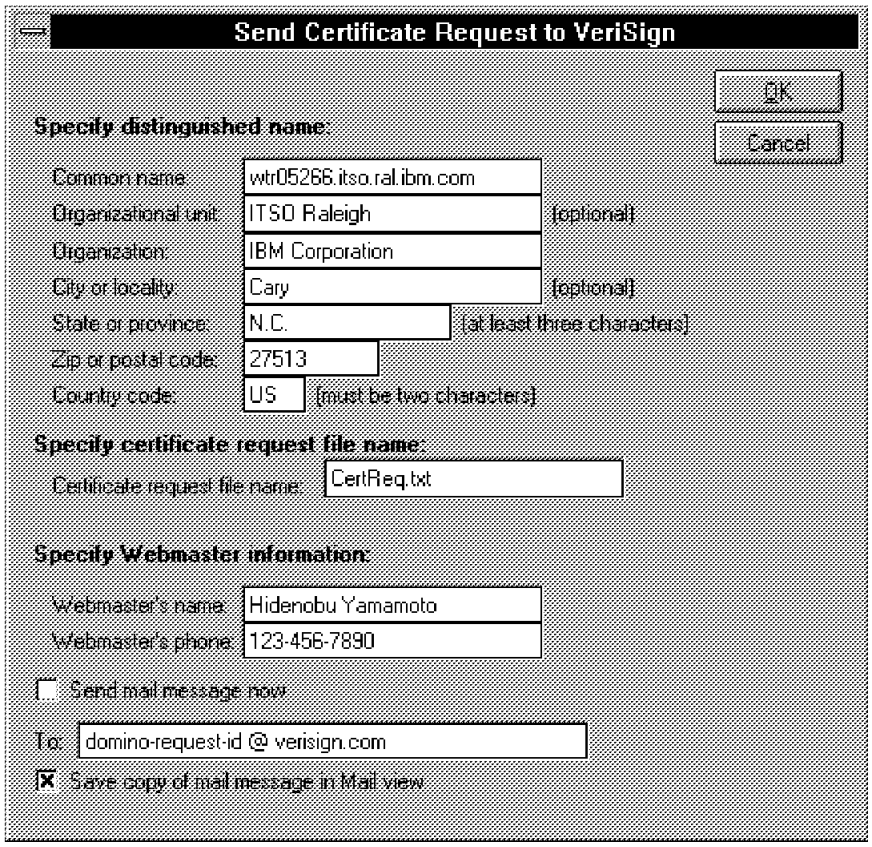


Figure 45. Send Certificate Request to VeriSign

Note: When you choose a CA other than VeriSign, you must supply a valid e-mail address of the CA.

The draft/backup copy of the mail message will be found in the Mail Backup view of this database (Figure 46 on page 66). To send this message to the recipient choose **Actions - Send Document**.

Figure 46. Certificate Request Mail Message Example

5. If you are setting up a commercial server, you will have to provide some additional, hardcopy, documentation at this point. Remember that by signing your public key certificate, the CA is saying to the whole world that you are an OK person to do business with. They will want some proof that this is so, such as financial and legal records of your organization.

6. After receiving the certified certificate by E-mail from VeriSign, go to Step 3: Merge certified certificate into key ring to store the certificate into the server's key ring file.

   Export the returned certificate included in the e-mail to a file. Click the **Merge certificate into key ring** button and supply the file name of the certificate (Figure 47 on page 67).

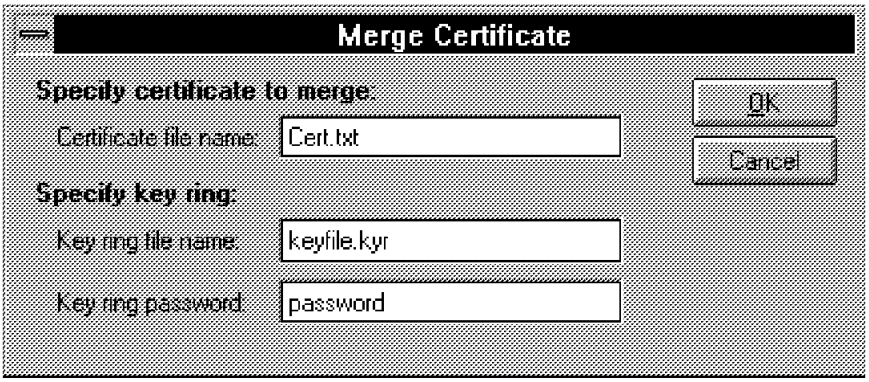


Figure 47. Merge Certificate for VeriSign

Note: When you choose a CA other than VeriSign, the CA will send an e-mail back to you that contains the certified certificate and a copy of the CA′s certificate. You must merge both certificates into the key ring.

Scenario 2: Running Domino as Internal CA: If you are setting up a Domino server for public access on the Internet, you will need a certificate from a well-known public CA, such as VeriSign. However, if your server is for *intranet* access, you may want to create your own certification hierarchy.

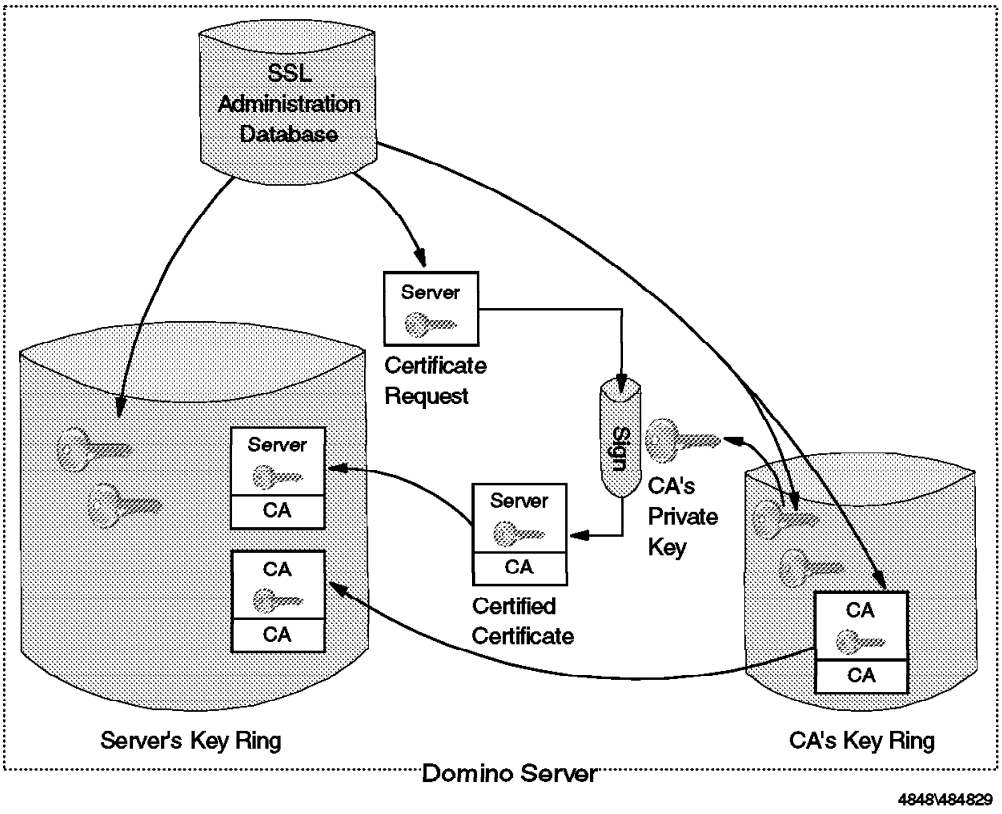


Figure 48. Overview of Scenario 2

Domino has the ability to create a CA key ring and to certify certificate requests; in other words, Domino itself can be run as a certification server or CA.

In this scenario, we:

1. Obtain a CA certification and CA key ring on the Domino server.
2. Create the server's certification request and key ring.
3. Certify the server's certification with the CA key ring.
4. Merge the server's certified certification in the server's key ring.

Please note that we assume you want your Domino server to have the ability to certify other servers. If you just want to test the SSL function of Domino, you should take the menu option Create Key Ring - Self-Certification as a shortcut rather than following this scenario.

In the first phase we describe how to obtain CA certification and a CA keyring.

1. Open the **SSL Administration** database and click the picture of a keyring. Click **Certificate Authority - Create Certificate Authority Key Ring** to create a CA certificate and a keyring (see Figure 40 on page 61).

Figure 49. Certificate Authority - Create Certificate Authority Key Ring

2. Enter the required information. For example, in Figure 50 on page 69 we specify the common name as Redbook CA and our organization unit as ITSO Raleigh.

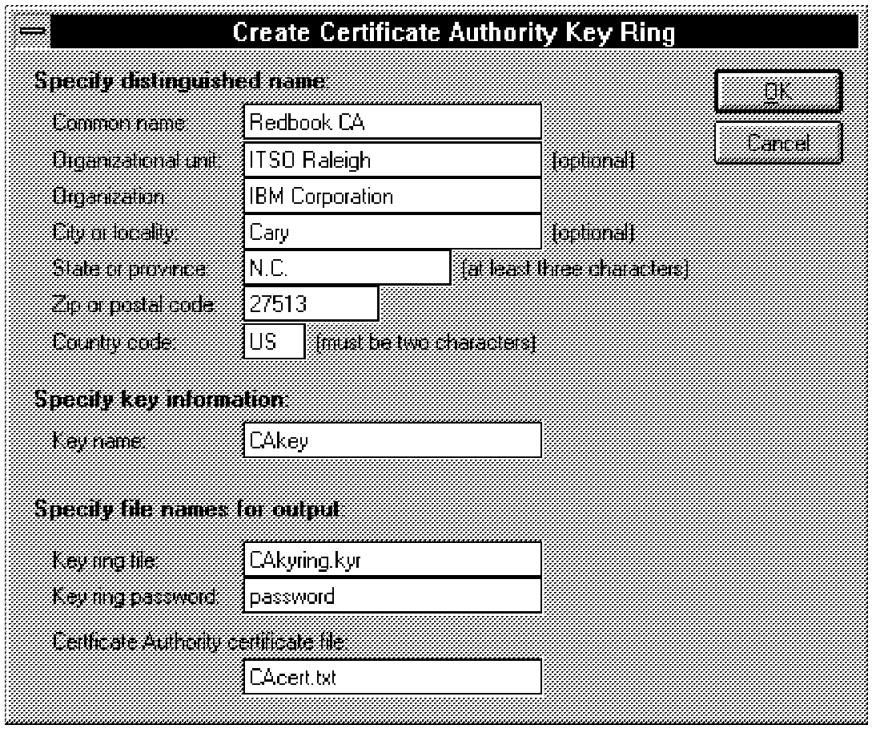


Figure 50. Create Certificate Authority Key Ring

Click on **OK** to proceed. After processing, the key ring file and the certificate authority certificate file are produced and placed in the Notes data directory.

Now Domino can be run as a root CA. You can certify user certificate requests using this certificate authority certificate. Any server or user can send a certificate request to the root CA as described in the previous section.

3. Go to the menu Create Key Ring - Internal Certification (Figure 40 on page 61) to create the server's key ring. Click **Step 1: Create a key ring and certificate request**. You don't have to do steps 2 and 3 this time.

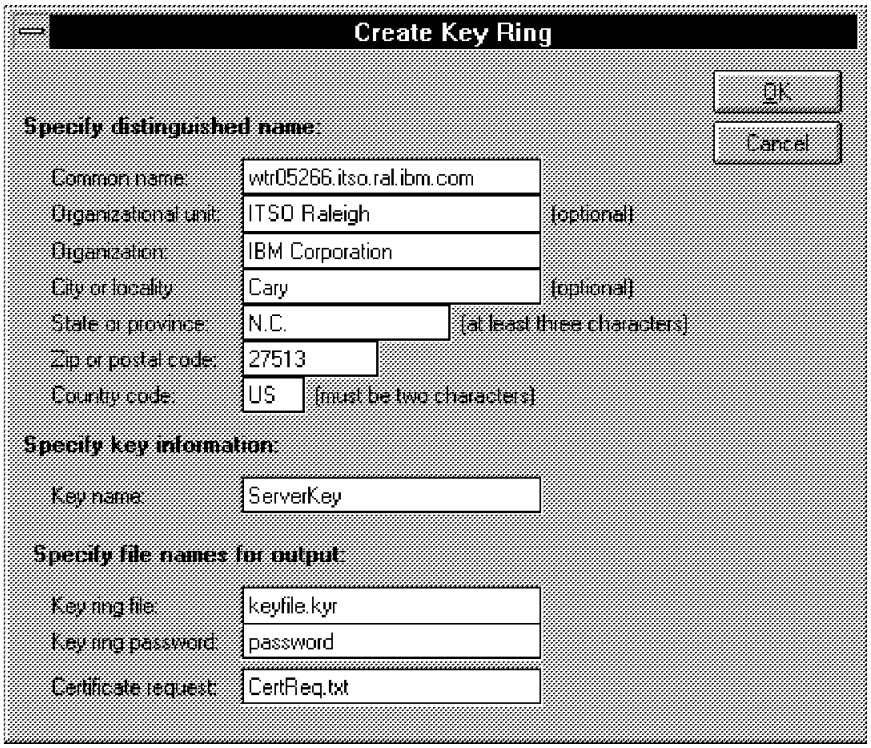


Figure 51. Creating the Server's Key Ring

4. Go to Certificate Authority - Certify Certificate Request in the SSL Administration Database menu (Figure 40 on page 61) to process the server's certificate request with your CA certificate.

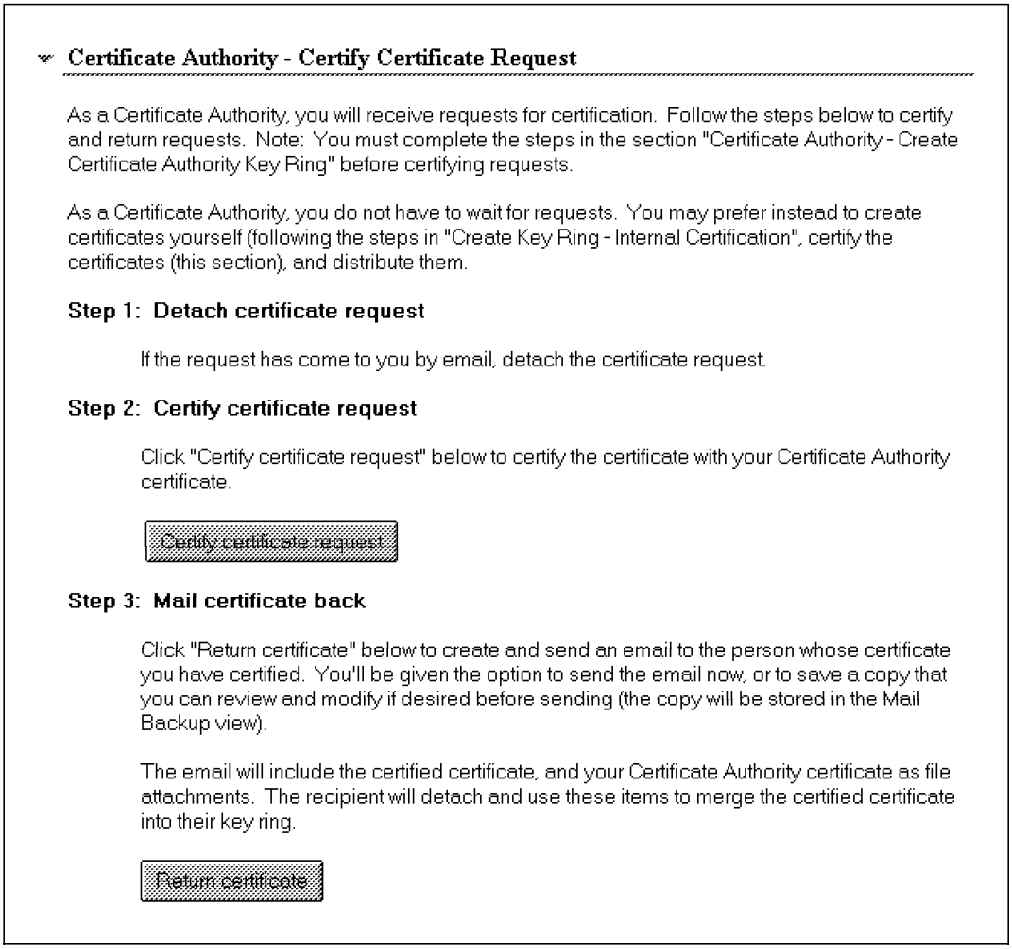


Figure 52. Certificate Authority - Certify Certificate Request

5. You already have the certificate request as a file (CertReq.txt in our example).

   Note: Usually you must obtain the certificate request as a file somehow. For example, if the request has come to you by e-mail, detach or export the certificate request from the message to a file.

6. Click on **Step 2: Certify certificate request** and supply the required information. Make sure the *incoming* certificate request file name, your CA key ring file name, and the certificate file name for the output are correct.

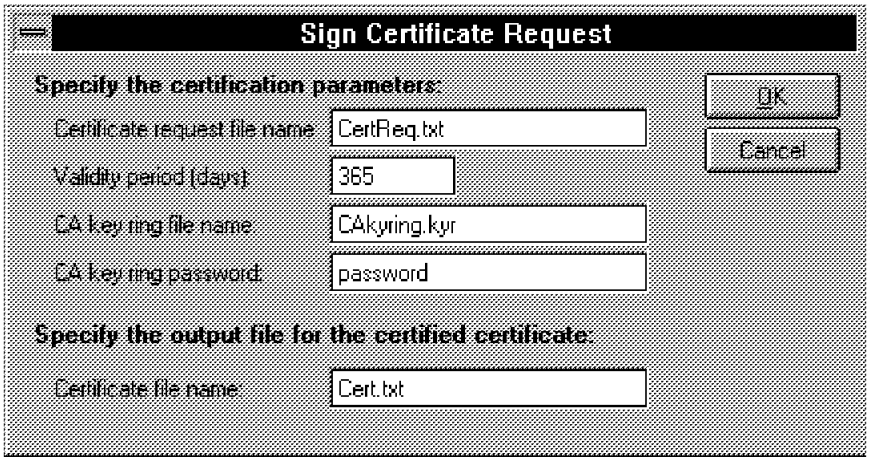


Figure 53. Sign Certificate Request

As a result, you will obtain a certified (or *signed*) certification as shown in Figure 54.

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 4,MIC-ONLY
Content-Domain:  RFC822
Originator-Certificate:
 MIIB/DCCAaYCBDJO1HEwDQYJKoZIhvcNAQECBQAwgYExCzAJBgNVBAYTAlVTMQOw
 CwYDVQQIEwROLkMuMQOwCwYDVQQHEwRDYXJ5MRgwFgYDVQQKEw9JQkOgQ29ycG9y
 YXRpb24xFTATBgNVBAsTDElUUO8gUmFsZWlnaDETMBEGA1UEAxMKUmVkYm9vayBD
 QTEOMAwGA1UEERMFMjc1MTMwGhcLOTYxMDI4MTUOMloXCzk3MTAyODE1NDJaMIGQ
 MQswCQYDVQQGEwJVUzENMAsGA1UECBMETi5DLjENMAsGA1UEBxMEQ2FyeTEYMBYG
 A1UEChMPSUJNIENvcnBvcmFOaW9uMRUwEwYDVQQLEwxJVFNPIFJhbGVpZ2gxIjAg
 BgNVBAMTGXdOcjA1MjY2LmlOc28ucmFsLmlibS5jb2OxDjAMBgNVBBETBTI3NTEz
 MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAM+39H7KvYmeBHUAU/iYXxgaTcSpd1bz
 XTwwdVfsUBv2GOUhpxbQcB5Cme+ppVxcrI/IUIfHhABUKbQxhXyClCsCAwEAATAN
 BgkqhkiG9wOBAQIFAANBAIWuhm1Wa3BEp1MqO/lJDgekCxEV47VFJ5OJ8vGeoyO5
 8mNJFiR7wzUvcXvZ7R76TuTl2OCsIttw7Jmdbokyprw=
Issuer-Certificate:
 MIIB7TCCAZcCBDJOhnYwDQYJKoZIhvcNAQECBQAwgYExCzAJBgNVBAYTAlVTMQOw
 CwYDVQQIEwROLkMuMQOwCwYDVQQHEwRDYXJ5MRgwFgYDVQQKEw9JQkOgQ29ycG9y
 YXRpb24xFTATBgNVBAsTDElUUO8gUmFsZWlnaDETMBEGA1UEAxMKUmVkYm9vayBD
 QTEOMAwGA1UEERMFMjc1MTMwGhcLOTYxMDI4MTAwOVoXCzk3MTAyODEwMDlaMIGB
 MQswCQYDVQQGEwJVUzENMAsGA1UECBMETi5DLjENMAsGA1UEBxMEQ2FyeTEYMBYG
 A1UEChMPSUJNIENvcnBvcmFOaW9uMRUwEwYDVQQLEwxJVFNPIFJhbGVpZ2gxEzAR
 BgNVBAMTClJlZGJvb2sgQOExDjAMBgNVBBETBTI3NTEzMFwwDQYJKoZIhvcNAQEB
 BQADSwAwSAJBANOsfTJqp8vqa1rclmMDfEXvmmd2diVDBTPRuSC8+RX5bDH68bCd
 s5FA9YX4QeR+lgsri/KG3YNBzRcSgUjXeYkCAwEAATANBgkqhkiG9w0BAQIFAANB
 AEliR8rjNmlm4lRaR6OFyhhTN5PkrO7cce5hWZdCrhVWO5iRCsAmJjHWjxywu3S1
 9a76zXaW9mR2ItIxaVxferA=
MIC-Info: RSA-MD5,RSA,
 qFPQ8xS/9pbS6/InSK9SHI12lun8762y+F8qh7g7TblnM5LHypb8J1QwKWYz5l5W
 xjxuupP4oKGbLBTh92C9fg==

VGhpcyBpcyBhbiBSRkMtMTQyNCBDU1IuCg==
-----END PRIVACY-ENHANCED MESSAGE-----
```

Figure 54. Certified Certificate Example

Note: Normally you would return the certificate to the person whose certificate you have certified. If you want to send it via e-mail, go into Step 3: Mail certificate back and click the **Return certificate** button.

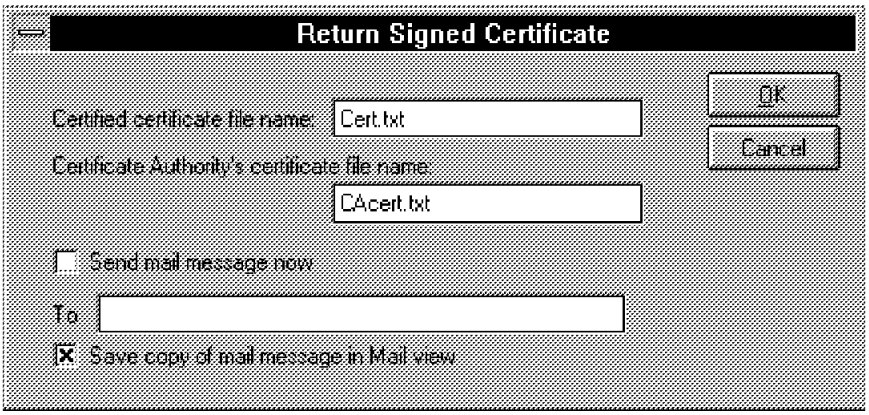


Figure 55. Return Signed Certificate

7. Merge the signed certificate and the CA′s certificate into the server′s key ring. Go back to Create Key Ring - Internal Certification and click **Step 3: Merge certified certificate into key ring**. Click **Merge certificate into key ring** button and enter the required information, then click **OK**. (Figure 56).

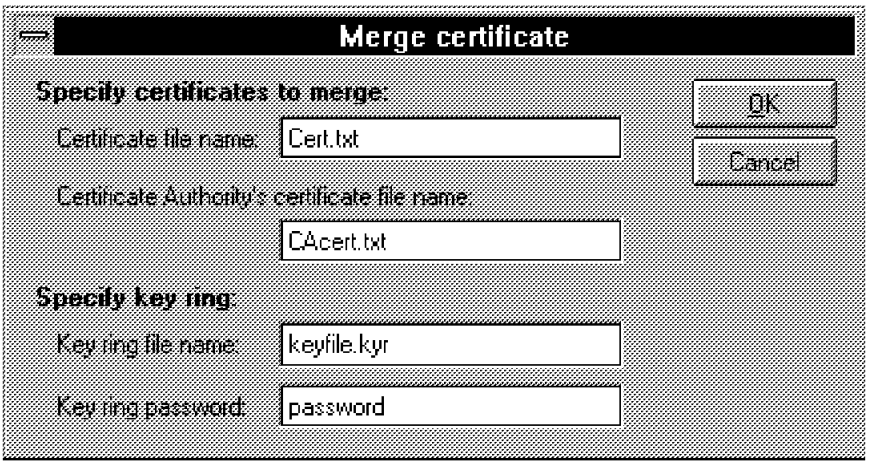


Figure 56. Merge Certificate for Internal CAs

5.2.2 Using SSL to Improve User Authentication

Our Web browser uses the SSL handshake mechanism to authenticate (or *trust*) the Domino server and establish a secure channel. However as we discussed in 4.1.2.5, “SSL Client Authentication” on page 46, SSL client authentication is not yet well established. Domino therefore uses the HTTP basic authentication scheme, based on user IDs and passwords to establish trust in the user.

Note: Domino Server 4.6 supports SSL 3.0, which verifies server and client identities.

As we described in 4.1.1, “HTTP Basic Authentication” on page 38, HTTP passwords are not strongly protected. If we want to be able to rely on them, we need to encrypt the messages containing them, using SSL. Figure 57 on page 74 shows the logical perspective of the two directions of the authentication. The user trusts the server′s certificate using the CA′s certificate to create a secure channel between the browser and the Domino server. Then Web basic

authentication is performed (username and password are sent to the server) so that the Domino application may trust the user.

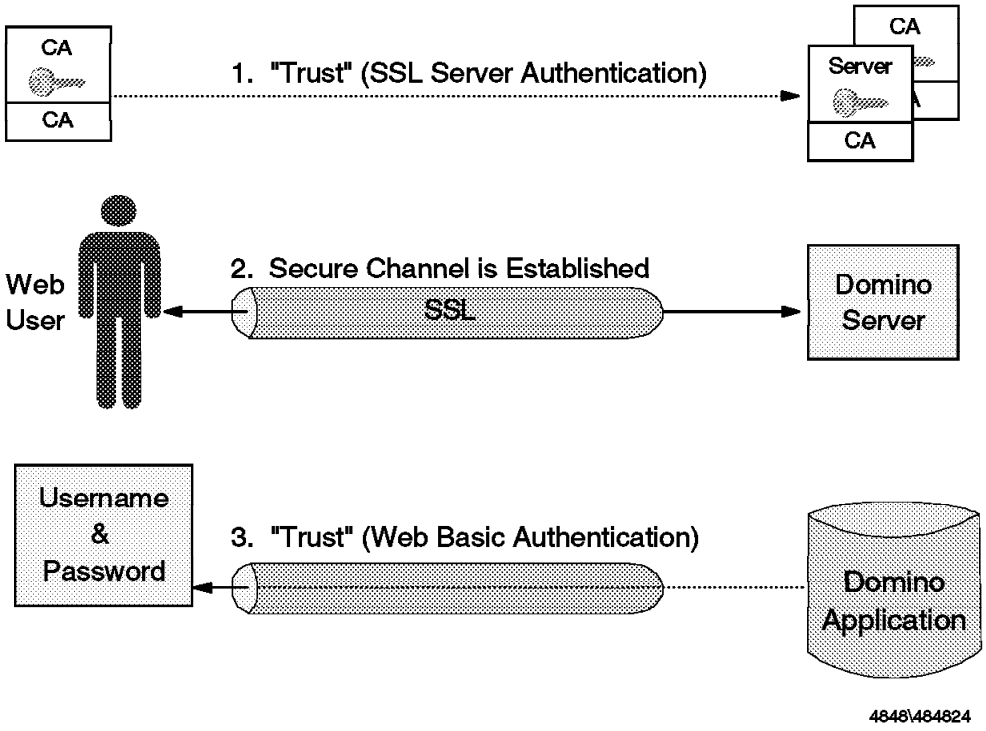


Figure 57. SSL Server Authentication and Web Basic Authentication

If you refer to 4.1.1, “HTTP Basic Authentication” on page 38 you will see that the user ID and password are sent with *every* request for an access-controlled URL, even though the user is only prompted to provide them the first time they are needed. Application developers should keep this in mind if they are using SSL to protect the password. It is important to maintain the SSL channel for all requests, not just the initial database access. Otherwise an attacker can capture the data packets and extract the user ID and password (Figure 58 on page 75).

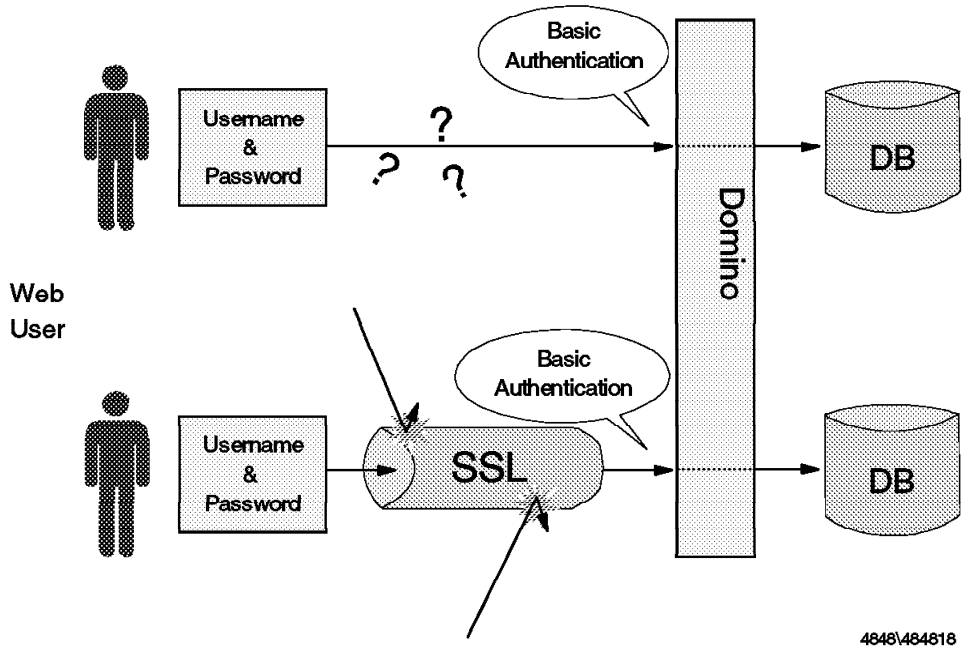


Figure 58. Basic Authentication with or without SSL

5.2.3 Selective Use of SSL Based on User ID

HTTP user IDs and passwords are frequently used as a method of tracking a particular person, rather than as a true authentication method. In this case the exposure of the password does not cause any loss of access control, so it may not be appropriate to protect it with SSL. You should bear in mind that SSL uses heavy encryption algorithms which carry a significant cost in extra CPU processing. (Studies have shown CPU utilization figures are 12-20% higher for SSL compared to normal HTTP.) You may want to turn SSL on or off according to the user's privilege.

5.2.3.1 Simple Example - Frontdoor Database

We illustrate our point with a simple example. Our frontdoor database classifies incoming user login requests into three groups (Figure 59 on page 76):

Authenticated and Encrypted

Highly privileged users may have access to confidential data. The Web session must be authenticated and encrypted. That is, SSL must be running throughout the session.

Authenticated but Not Encrypted

In cases where the username works just as a tag and the user doesn't have access to confidential data, the Web session should be authenticated but not be encrypted. That is, SSL is not required after the authentication process is done.

Not Authenticated and Not Encrypted

When the data is opened as a public domain, both authentication and encryption are not required.

Figure 59 on page 76 shows the three login routes.

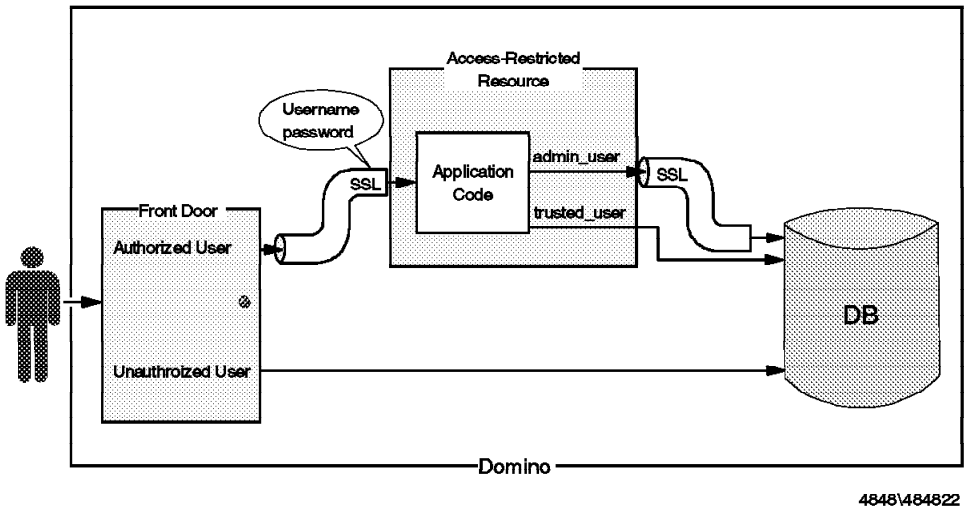


Figure 59. Overview of FrontDoor Example

In this example, two components play an important role. The first is the Welcome Page and the second is the Access-Restricted Form.

The Welcome Page Navigator is the first component of our Frontdoor example. It classifies the user into two categories: users to be authenticated and users not to be authenticated (Figure 60).

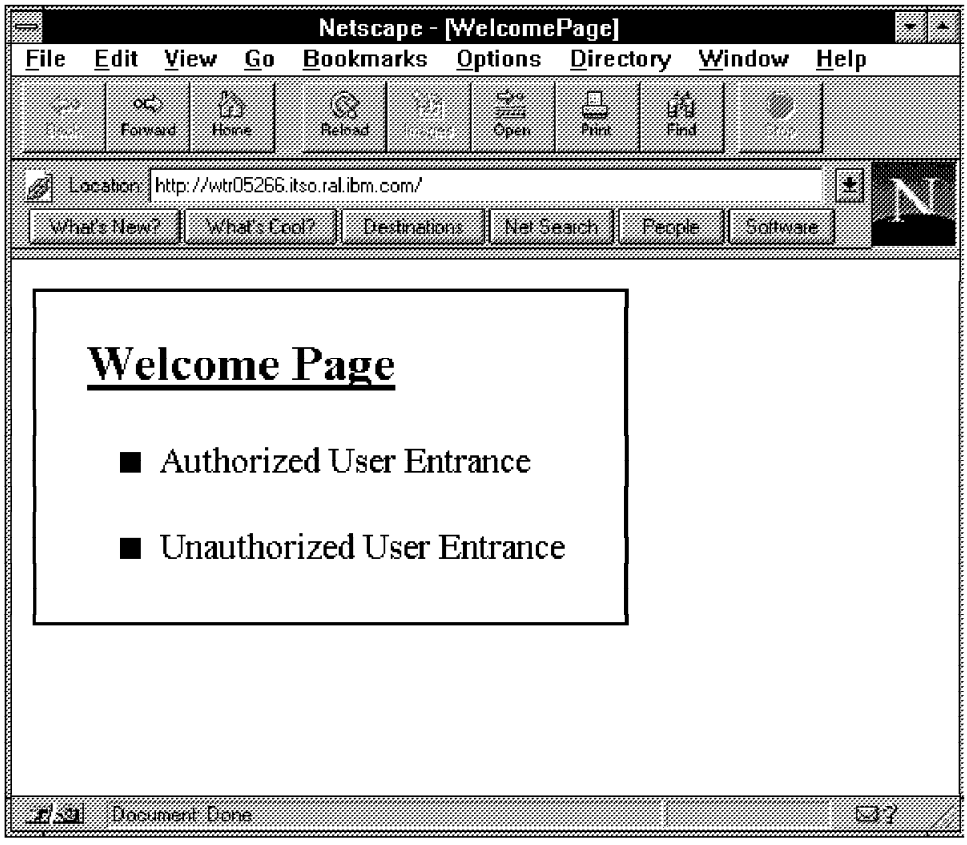


Figure 60. Welcome Page of Frontdoor Database

The Access-Restricted Form, the second component of our front door example, classifies the authenticated users into two groups: users who need SSL and users who no longer need SSL (Figure 61 on page 77). Inside the form, a formula will examine the username and return a URL in response to selecting the submit button (Go in Figure 61).

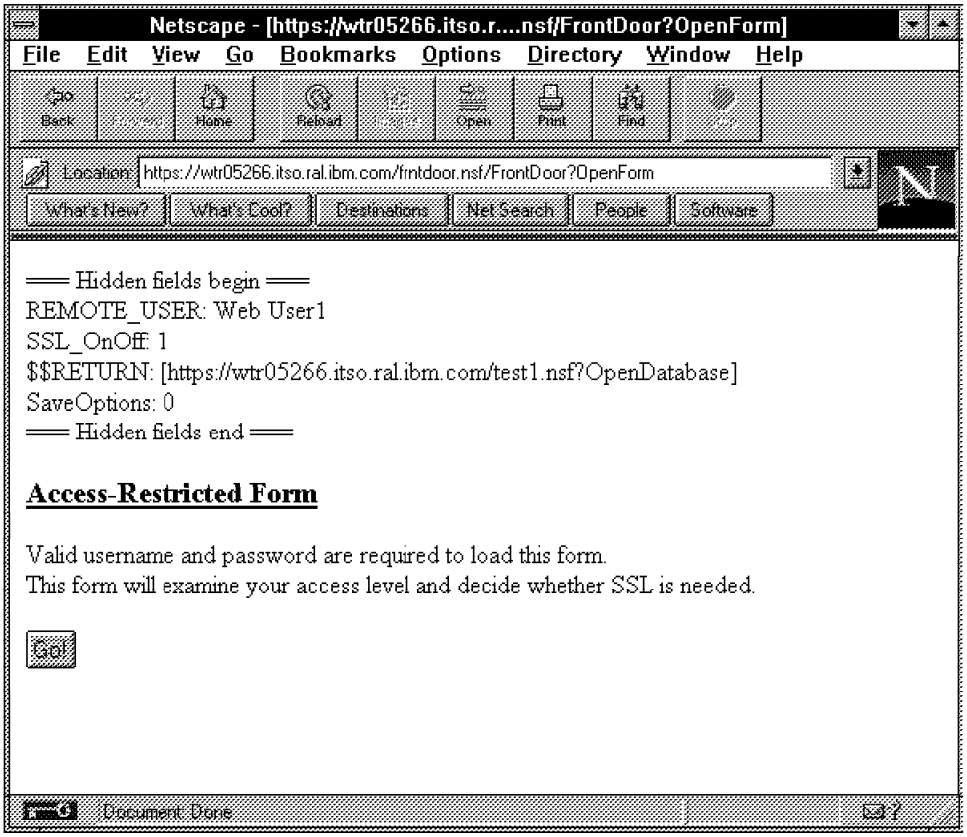


Figure 61. Access-Restricted Form of Frontdoor Database

Note: You can see "hidden fields" in Figure 61. We did not hide these fields at this point because we want to illustrate the behavior of the FrontDoor form. You can change the attributes of these lines in the application database to hide them from the user.

In this example the Frontdoor application is a separate database, but in general you would want to merge it with the application that it provides access to. The reason is that otherwise it is possible for a user to bypass SSL by simply entering an http:// URL in place of https://. Merge the Frontdoor and your application and you will have a single *entry point* as seen in Figure 62 on page 78. We created a sample database for our project that includes this technique. The details can be seen in Chapter 6, "An Example Application" on page 101.

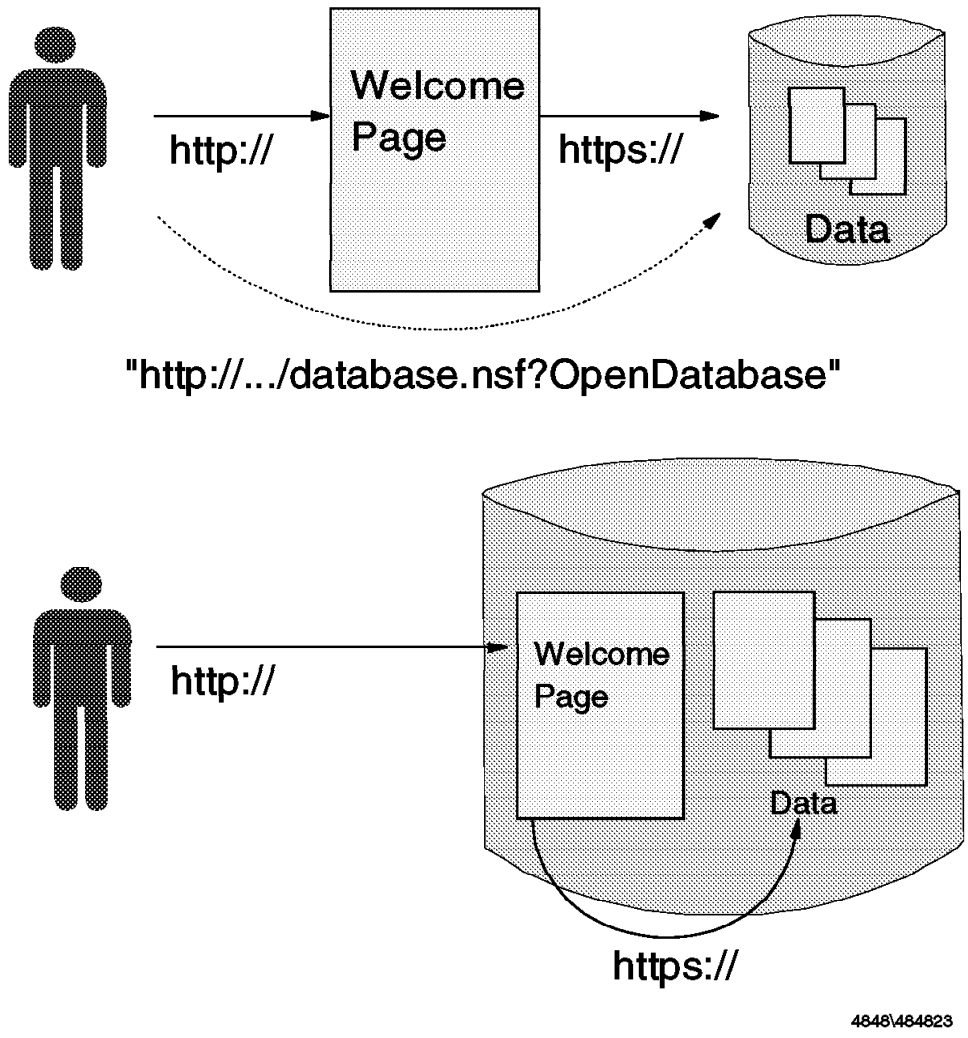


Figure 62. Merging Frontdoor and Application Databases

5.2.3.2 Details of the Frontdoor Application

The Frontdoor application selects the SSL state by placing URL links embedded behind menu items. Another way is to send an HTML command which leads the user to follow a specified URL link (see 5.2.3.4, “Turning On/Off SSL According to Username” on page 82).

To continue with our Frontdoor example, we used a navigator as an entrance (Figure 63 on page 79). You can design a navigator to be translated by Domino into a clickable map in a Web page.

Note: The About database document also can be used to implement this function.

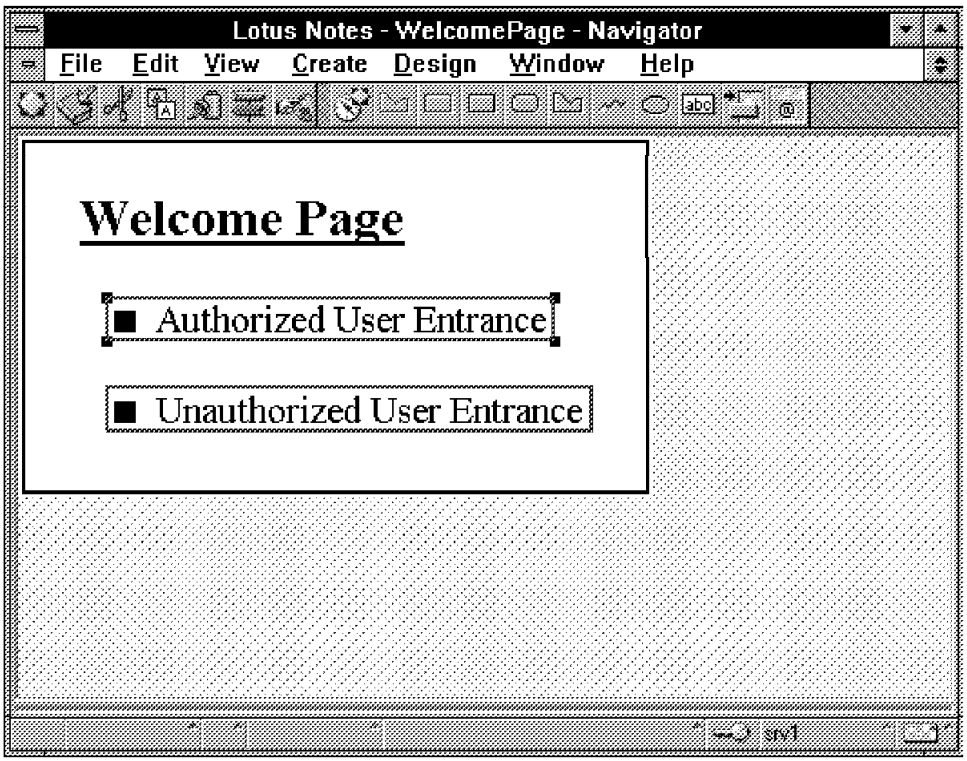


Figure 63. Designing Frontdoor Navigator

Two menu items exist in our sample navigator. One is for registered Web users and another is for unregistered (Anonymous) Web users.

You should configure the database properties infobox so Web users see this navigator as the initial screen of the database (Figure 64). You can also configure the server document so that Domino loads this database as the initial screen (Figure 65 on page 80).

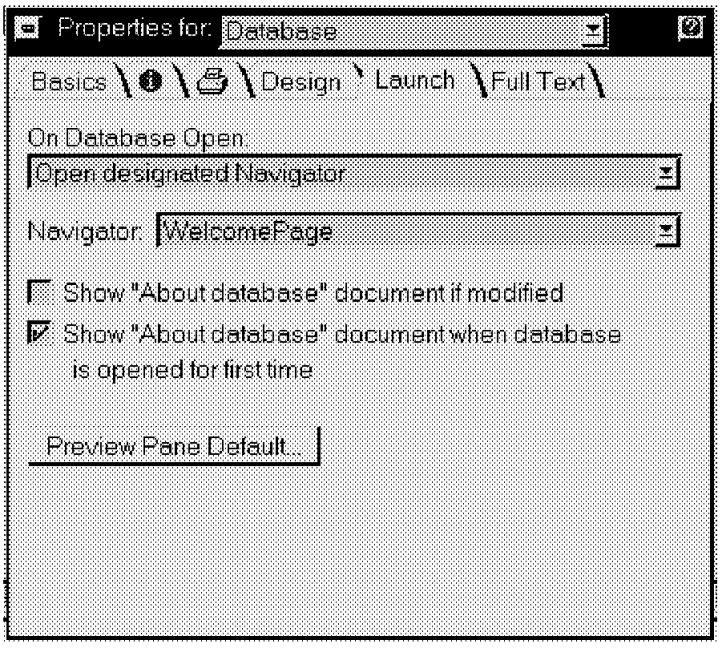


Figure 64. Launching the Navigator

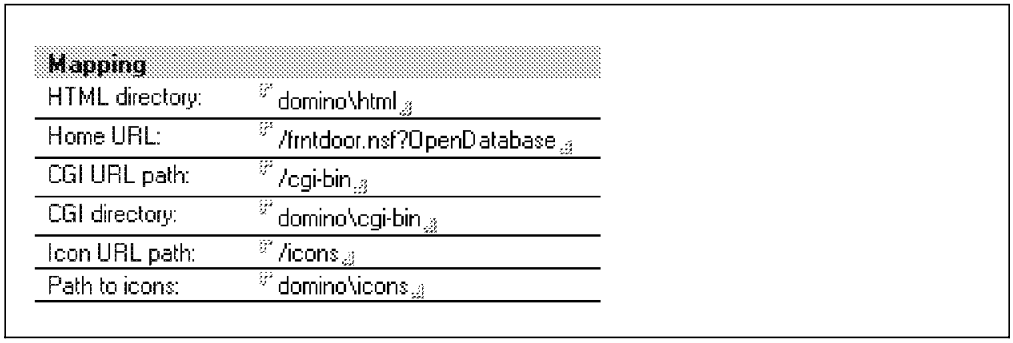


Figure 65. Server Document Settings

Inside this navigator two URL links are embedded. The first one is for registered users and points to an access-restricted form. SSL is used because we used https: instead of http: in the URL (Figure 66). The other is an ordinary URL link to open a destination database (Figure 67 on page 81).

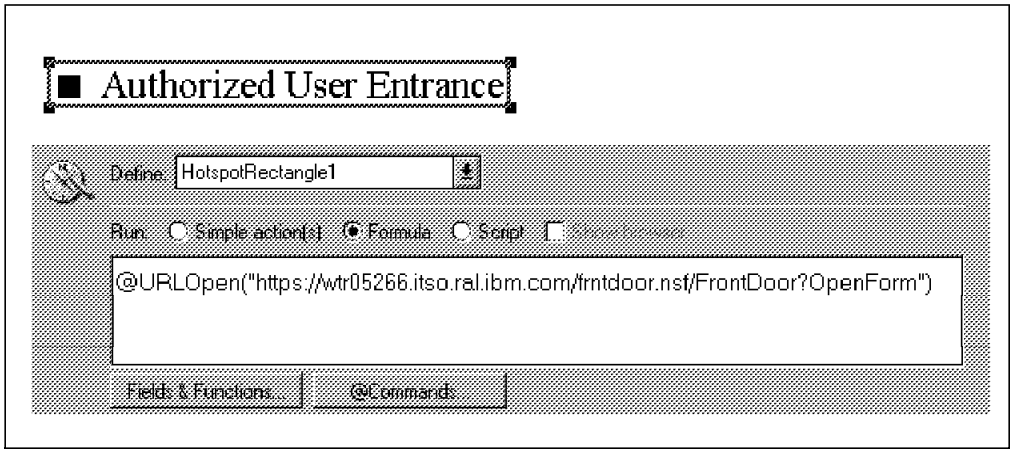


Figure 66. Secure URL Link to the Access-Restricted Resource

Note: In Figure 66, frntdoor.nsf is the file name of the database and FrontDoor is the name of the form. You may use the formula @Command([Compose]; ″FrontDoor″) to open the specified form instead of @URLOpen if you do not need SSL. When Notes builds URLs from Notes @Commands the URL will be translated as http:// or https:// according to the previous URL link. If the user reached this navigator using an http:// link, Domino translates this Notes link into http://. If the user reached this navigator with an https:// link, Notes will use https://.

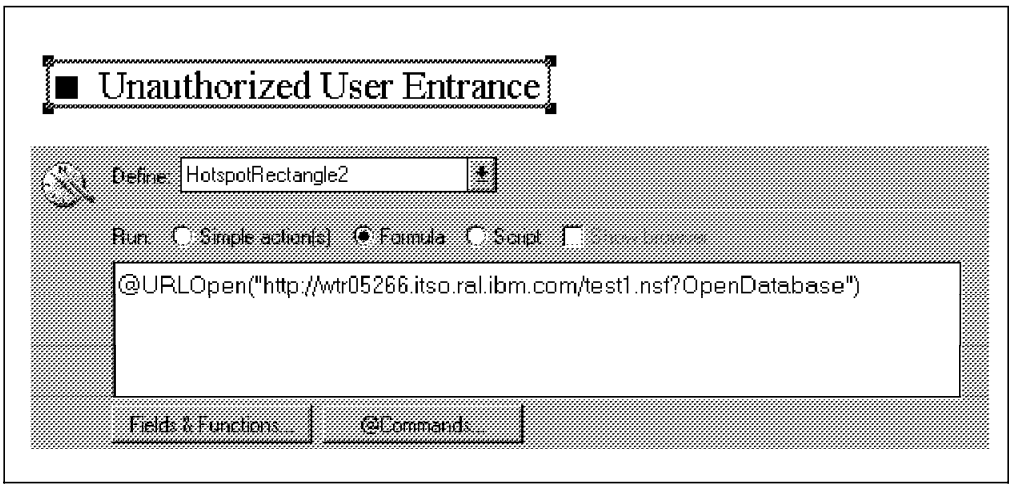


Figure 67. Ordinary URL Link

5.2.3.3 Requesting Username and Password

You can control the point where the user should enter the username and password. Using this technique with SSL, the application can authenticate the user securely.

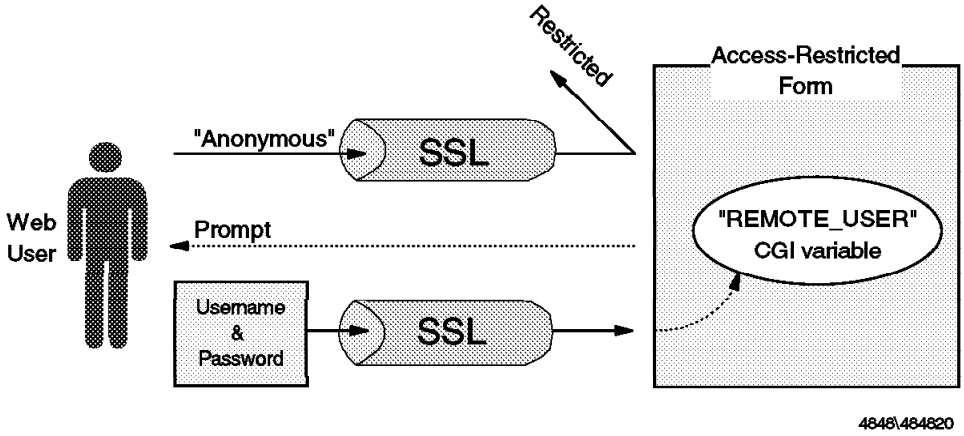


Figure 68. Requesting Username and Password

Create a URL link that points to an access-restricted resource, such as a database or form, where you want the user to enter a username and password (Figure 66 on page 80). When a user selects the link and doesn't have proper access rights to it, basic authentication will be performed and the user is prompted for the username and password (Figure 69 on page 82).

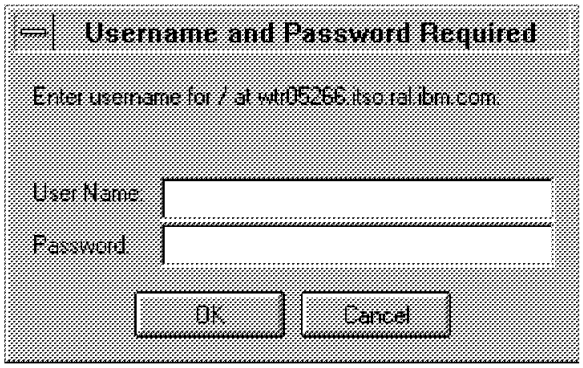


Figure 69. Prompt for Username and Password

This link should use https:// (secure URL) because the information should be kept secret at this point. After the authentication is done, the Notes application database on the Domino server may handle the username to perform appropriate action (see 5.2.3.4, “Turning On/Off SSL According to Username”). If SSL is not needed, Domino will turn it off.

5.2.3.4 Turning On/Off SSL According to Username

You can create a Domino application that depends on properties of the user ID. Our simple example examines the username and turns on or off SSL.

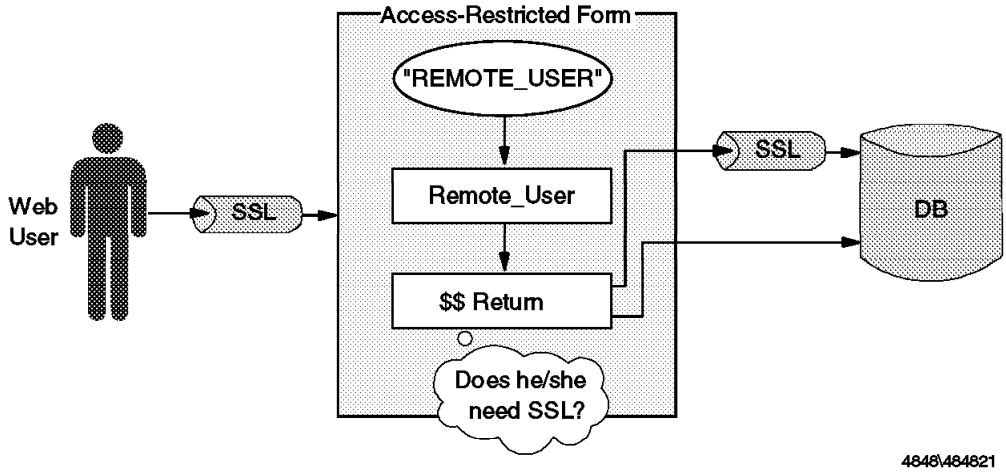


Figure 70. Turning On/Off SSL

In the Domino application database, the @Username function receives the username obtained from the basic authentication in the same manner as a pure Notes application. With Domino, Notes applications can also use CGI/HTTP environment variables inside the application database. A field that has the special name REMOTE_USER receives the value of the REMOTE_USER CGI environment variable, which represents the username.

Note: Notes agents can also use CGI variables.

In our example, the Access_Restricted Form (Figure 71 on page 83) receives the Web username, determines if the user needs SSL, and then loads the appropriate URL link.

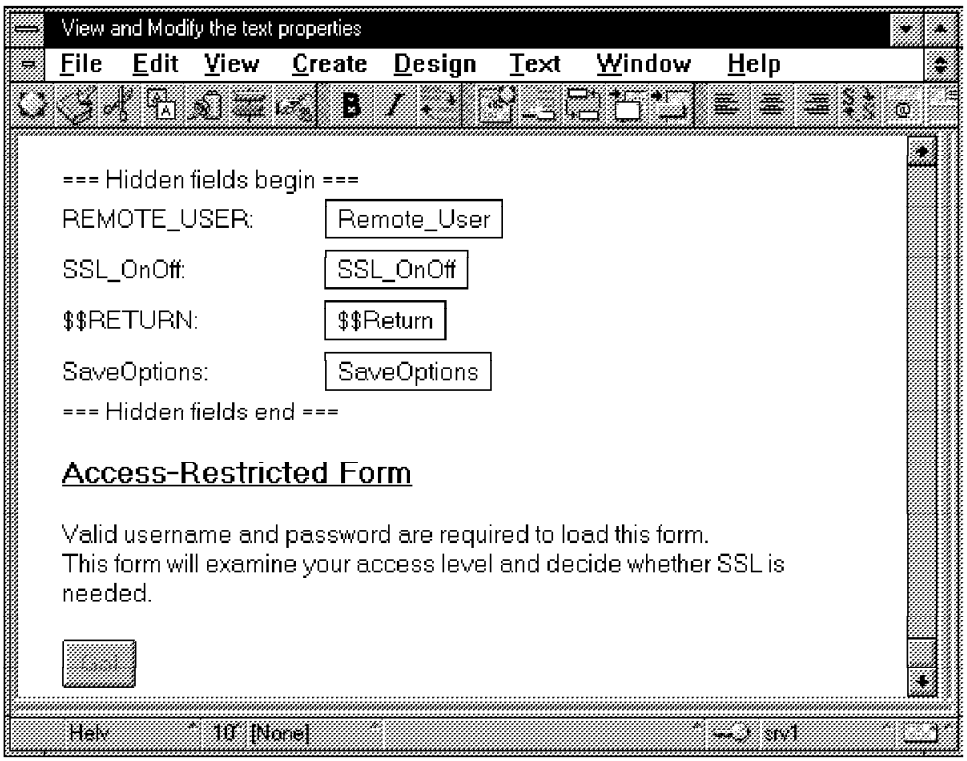


Figure 71. Design of Access-Restricted Form

Access Control: This form is access-restricted in order to force basic authentication. You can control the access in the form ACL or database ACL.

To open a form with ?OpenForm, the user must have access to create a new document using this form. On the other hand, to perform the authentication properly, Anonymous user access must be reader or lower to this form or database.

In this example, only Web User1 and Web User2 have access to this form (Figure 72).

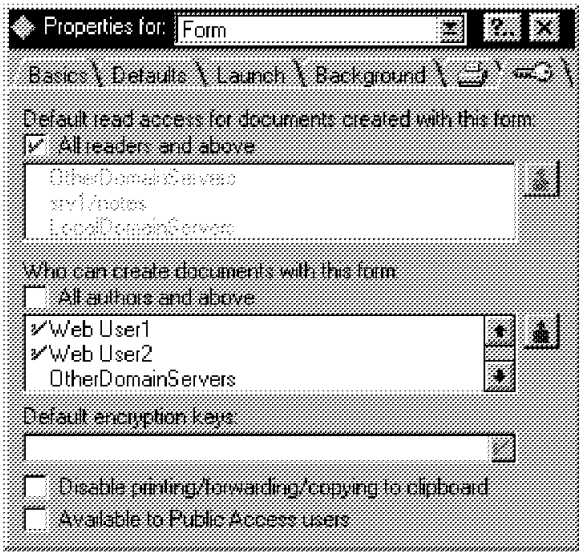


Figure 72. Access-Control of the Form

The fields on this form are “hidden fields” (see Figure 71). They can not be seen in read or edit mode.

REMOTE_USER Field: The REMOTE_USER field receives the REMOTE_USER CGI environment variable. This application uses the value of this field to determine whether or not the user should use SSL.

Note: You may use @Username instead of the REMOTE_USER field. @Username results in the same value as REMOTE_USER in a Domino application unless the user is Anonymous. In that case @Username returns Anonymous but REMOTE_USER doesn’t return any value.

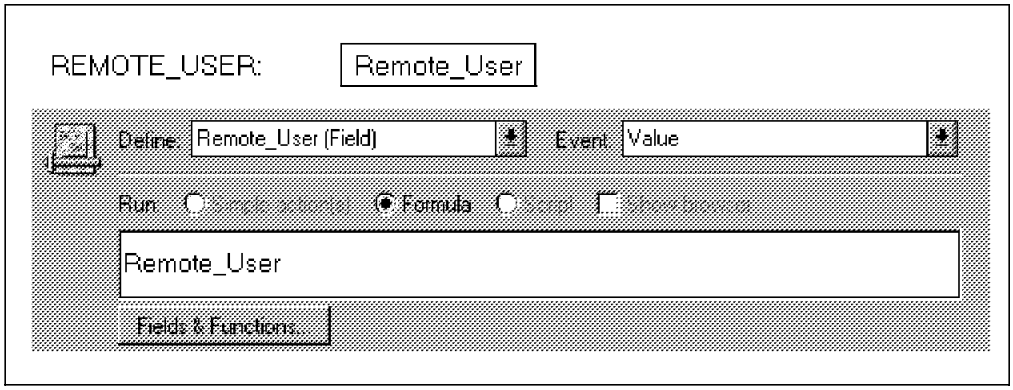


Figure 73. REMOTE_USER Field of Access-Restricted Form

SSL_OnOff Field: We added a field named SSL_OnOff, which represents whether SSL should be On or Off. In this simple example, the formula will examine if REMOTE_USER is Web User1 then result in 1 (on) or 0 (off). This is not a special field. We created this field to separate the user recognition logic from the $$RETURN field.

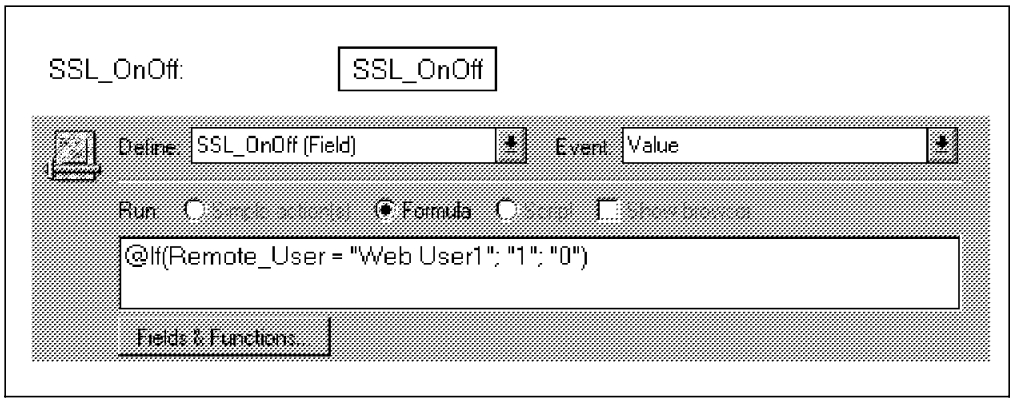


Figure 74. SSL On/Off

$$RETURN Field: The field named $$RETURN is a special field to Domino applications. This formula will be evaluated when the user clicks on the submit button. Using this field, the application can send any HTML statements or invoke CGI scripts when the button is selected.

Note: In a Domino application, a button in a form is recognized as a submit button. The formula in the button is ignored by Domino.

In this example, when the user clicks the button he/she will see the initial screen of the destination database TEST1.NSF, with or without SSL. SSL will be activated for Web User1 but not for Web User2.

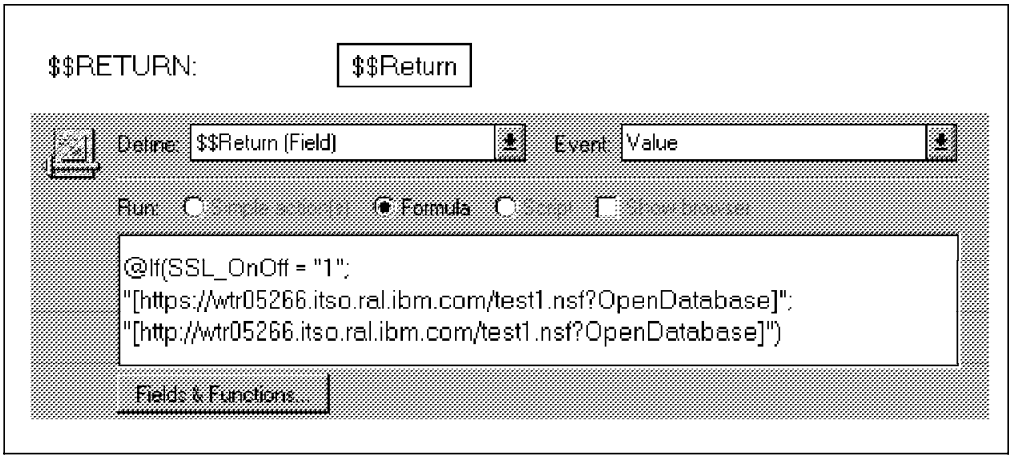


Figure 75. $$Return Field to Open Target Database with or without SSL

SaveOptions Field: The field named SaveOptions is a special field also. If this field exists in the form and the value is 0, the document that is composed using this form will not be saved into the database. This function is useful when the form and the submit button are used for communicating with a user and not for creating documents.

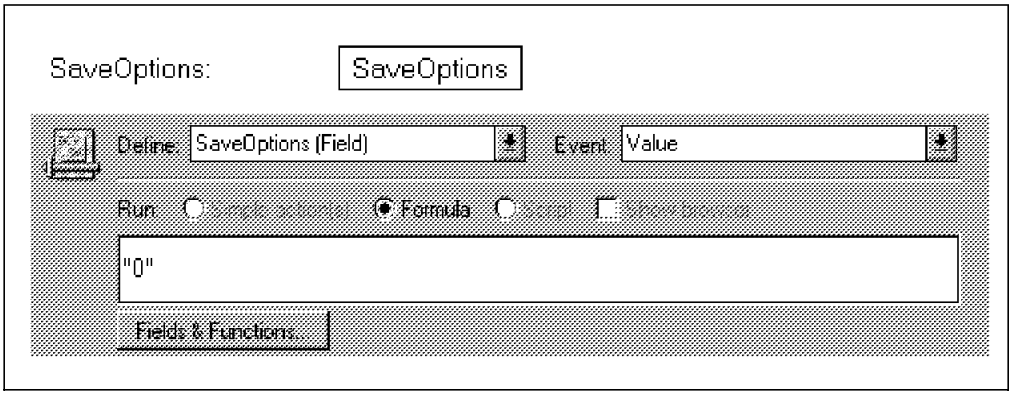


Figure 76. Using the SaveOption Field

As the result of our Frontdoor example:

1. A Web user will be authenticated when he/she attempts to go through the authorized user entrance.
2. If the user is Web User1 or Web User2, the attempt will succeed.
3. When the user clicks the button on the form, SSL will be activated for Web User1 and not activated for Web User2.
4. The user can open the destination database with or without SSL properly according to his/her access level.

5.3 Protecting the Domino Server

If you run a Notes server inside an enterprise network you do not usually have to think too much about how it may come under attack. By contrast, a Domino server providing access to Web users over the Internet is in a very exposed position. In fact, it has to boldly go where no Notes server has gone before. It may not come under attack by Klingons, but it *will* come under attack.

There are several aspects to consider when protecting a server in this situation:

- How to place it within a firewall so that it gets the best possible protection, but can still do its job. We discuss firewall aspects in Chapter 7, “Lotus Notes Firewall Configurations” on page 121.
- How to set up the Notes access controls so that the server only allows the access we want, and does not expose information about Notes objects that we want to hide.
- How to harden the underlying operating system, so that it does not offer a target to a would-be cracker.
- How best to set up logging and monitoring.

5.3.1 Controlling Access to Notes Resources

The database ACL is the first and the most important defense. To protect your Domino server, you must design the ACLs of all databases very carefully. Remember, the server access list does not work in Domino, but you can inhibit Web users from browsing the database list with the /?Open Domino URL. This is not security in a strict sense, but it does discourage the probes of an attacker.

Note: For ordinary HTML documents on most Web servers, including a Domino Web server, directory protection works by placing protection directives in the configuration file (file HTTPD.CNF in the case of Domino). On the other hand, for the application databases on Domino, we have to use Notes functions for protection.

5.3.1.1 Prohibiting Anonymous Web User Access to Domino Servers

You can prohibit anonymous Web user access to a Domino server (see Figure 77 on page 87).

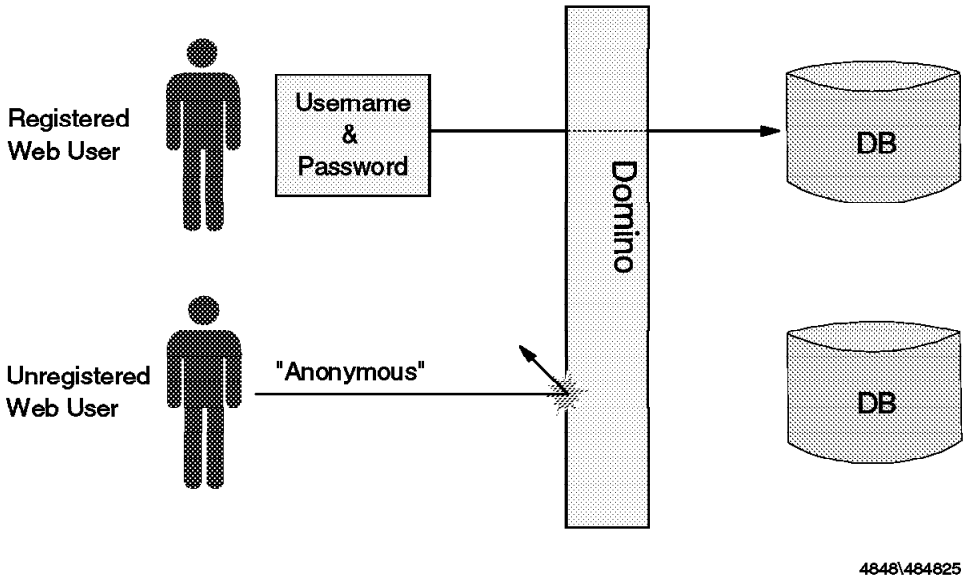


Figure 77. Prohibiting Anonymous Web User Access

There is a field named Allow anonymous HTTP connections in the security section of the server document (see Figure 78).

Security settings	
Compare public keys against those stored in Address Book:	Yes No
Allow anonymous Notes connections:	Yes No
Allow anonymous HTTP connections:	Yes
Allow HTTP clients to browse databases:	Yes
Check passwords:	Enabled Disabled

Figure 78. Security Section of Server Document

When this field is set to Yes, Web users can access databases as Anonymous users. When this field is set to No, every Web user will be asked to supply the username and password when he/she connects to the Domino server. No matter what database or form they request, they must have a valid username and password in the Public Address Book on the Domino server to gain access.

5.3.1.2 Protecting Databases from Anonymous User Access

To protect databases from unregistered (Anonymous) Web users, create an entry in each database's ACL called Anonymous and grant it No Access.

In a database ACL, Anonymous can be listed as an individual or as a member of groups or roles, the same as registered Web users. An entry named Anonymous in the ACL specifies the access level of unregistered Web users (Figure 79 on page 88). If Anonymous isn't listed in the ACL, Domino grants the unregistered Web user access based on the database's default access level.

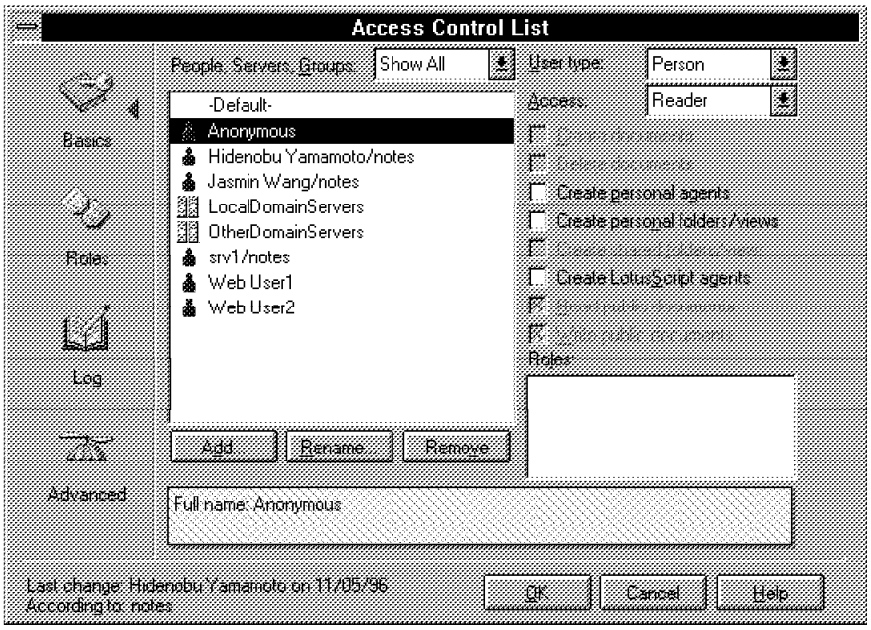


Figure 79. Using Anonymous User Entry in Database ACL

You should place an Anonymous entry in all database ACLs on the Domino server. In a mixed Notes and Domino environment a database administrator may assume the Default entry is for representing a Notes user. Anonymous users should be more restricted than that.

5.3.1.3 Maximum Internet Browser Access

The database ACL gives you the option to set an access level for Web users that will set an upper limit to Web access to the database, regardless of individual ACL settings. In the database ACL advanced panel, the field Maximum Internet Browser Access can be set to determine the maximum access allowed for the database from Web browsers.

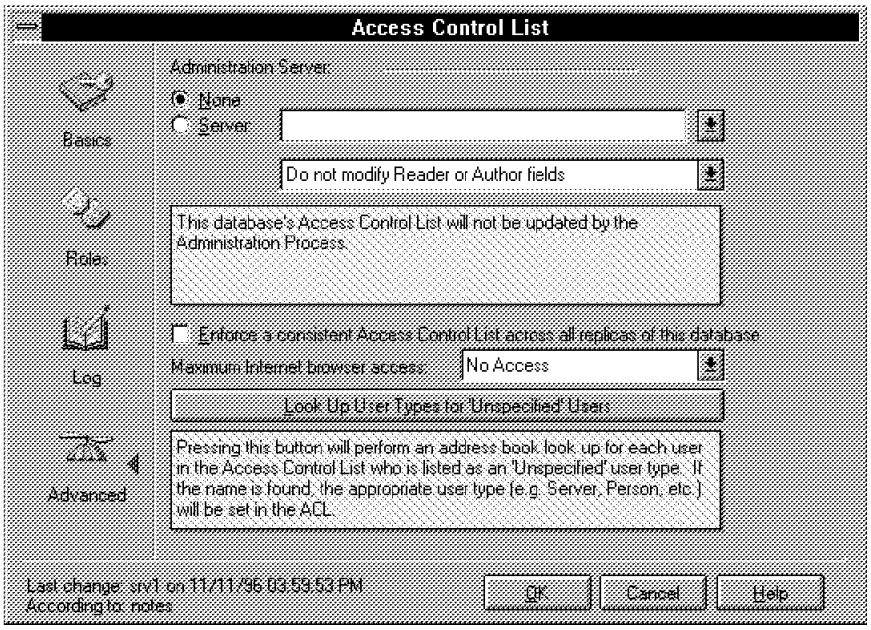


Figure 80. Specifying Maximum Internet Browser Access in Database ACL

5.3.1.4 Inhibiting Web Users from Browsing the Database List

You can inhibit Web users from browsing the database list. You usually want to do this, because in general you want Web users to follow the dialogs that you have carefully constructed and not be able to link to databases directly.

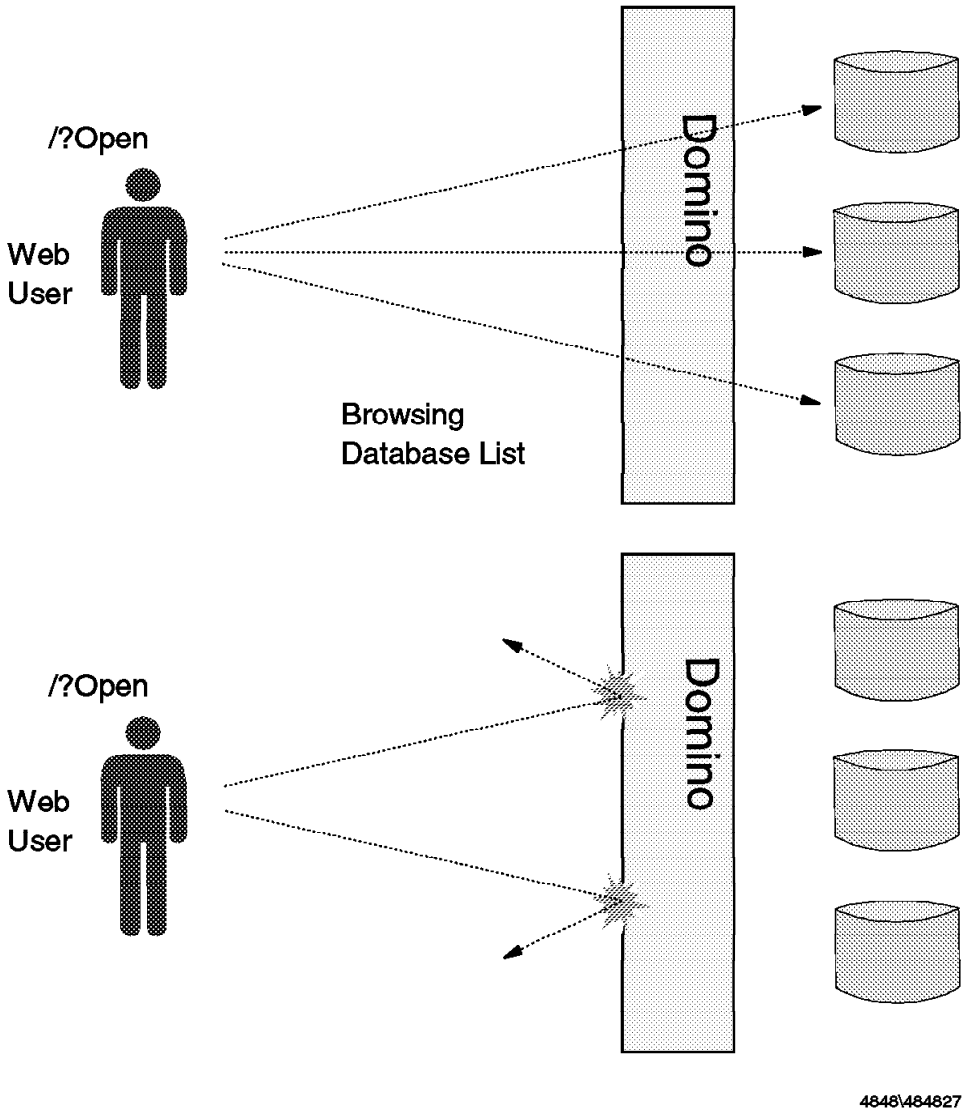


Figure 81. Browsing Database List

In the security section of the server document, there is a field named Allow HTTP clients to browse databases (Figure 78 on page 87).

When this field is set to Yes, Web users can browse the database list of the Domino server by using a Domino URL, such as http://domino.server.hostname/?Open (see Figure 82 on page 90). When this field is set to No, Web users cannot do that (see Figure 83 on page 91).

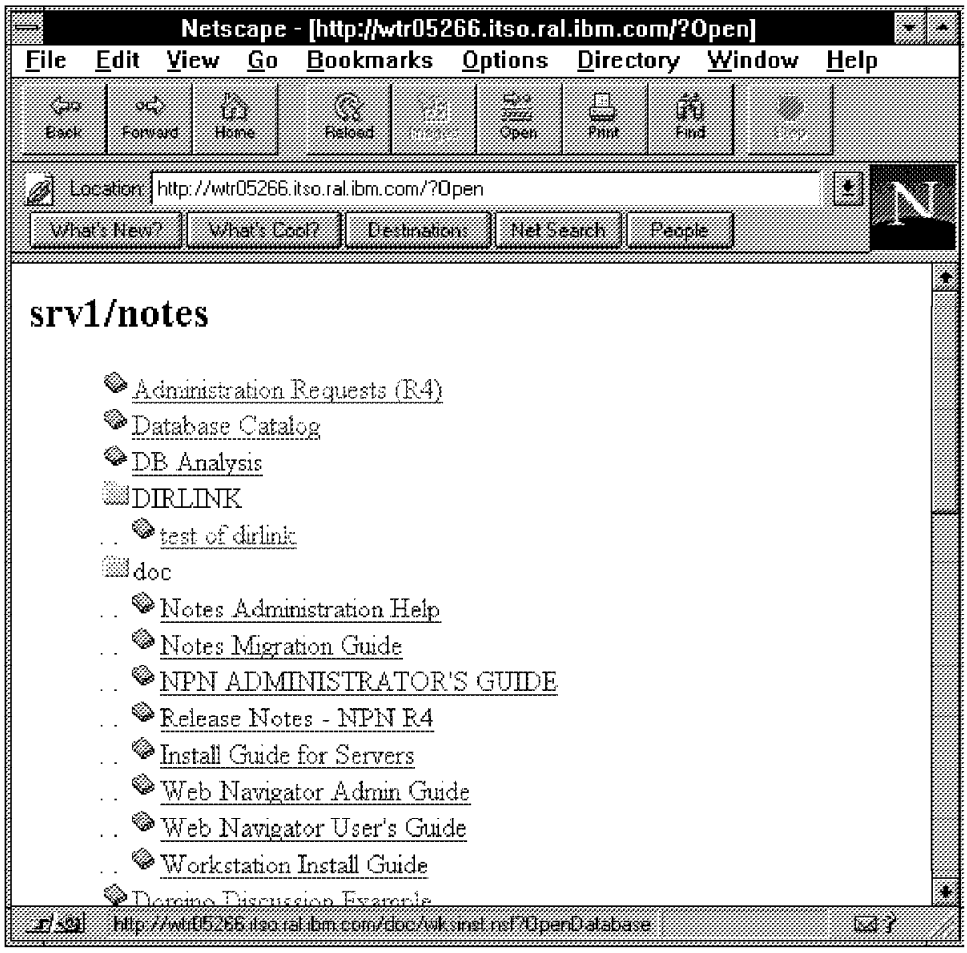


Figure 82. Result of /?Open When Database Browsing Is Allowed

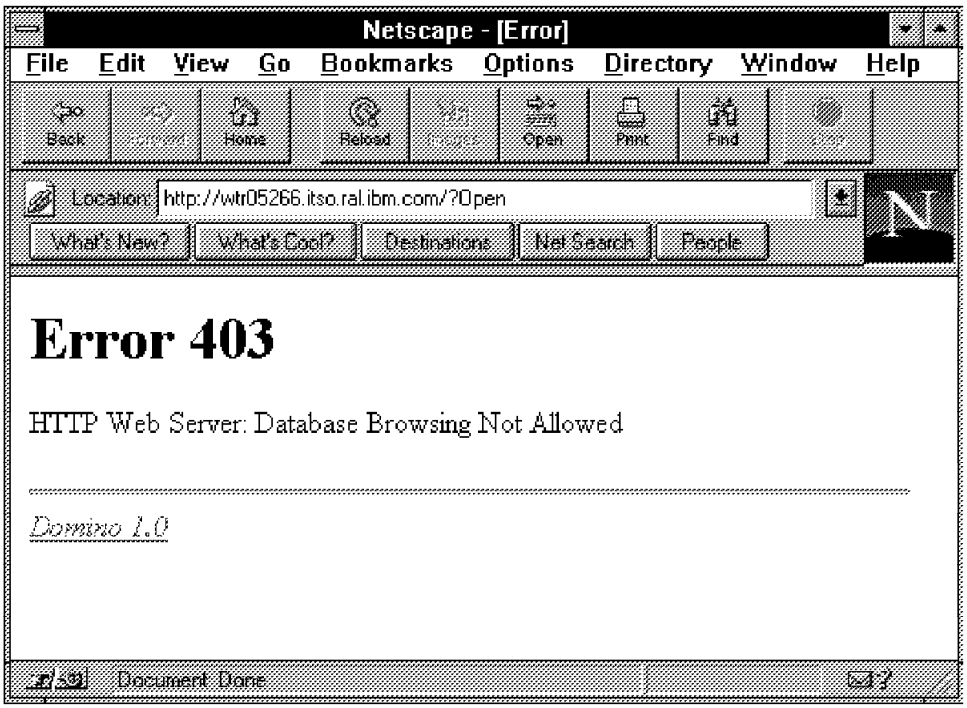


Figure 83. Result of /?Open When Database Browsing Is Not Allowed

5.3.1.5 Hiding a Database Name from the Database List

An alternative approach is to allow users the option of displaying the database list, but to hide databases that you do not want them to know about.

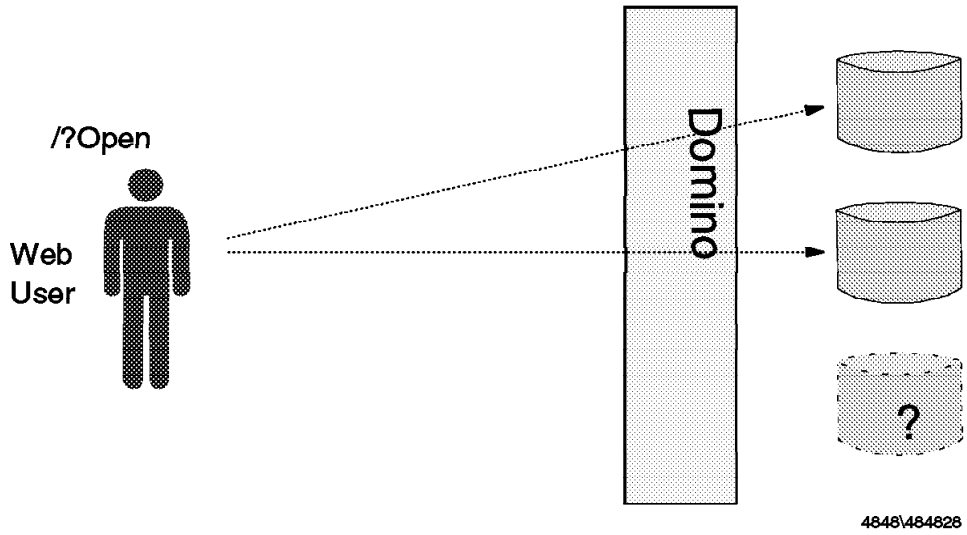


Figure 84. Hiding Database Name from Database List

You can hide a database name from the database list by clicking on the Show in Open Database dialog check box in the database properties infobox (Figure 85 on page 92). If you remove the check from that check box, the name will not be shown in database list from either a Web or Notes client. Note that if a user

knows the filename, he/she can access the database. Therefore, this is not an effective security measure.

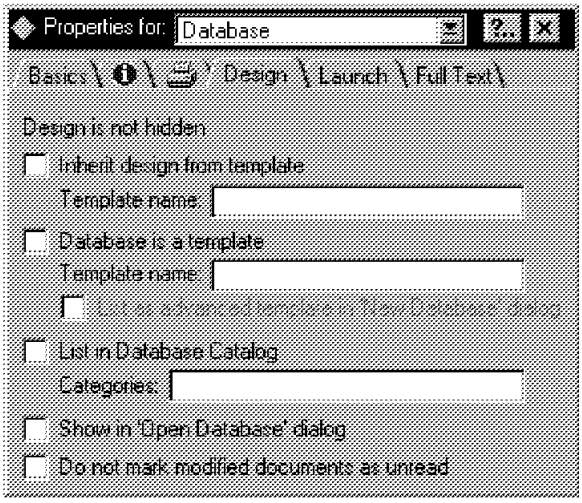


Figure 85. Hiding Database Name from Database List

5.4 Logging and Auditing

Logging is an important tool for detecting attempted attacks and retracing successful ones. Generally the base operating system provides some level of logging, such as the UNIX syslog function or the Windows NT system logs. Ideally you want to save the logs on some machine other than the exposed server itself. The reason for this is that if someone breaks into the system he or she may try to cover his or her tracks, by removing log records, etc. The replication capabilities of Notes give us a simple way to achieve this. In addition to system logs, the Domino server provides a number of other logging facilities, which we focus on here.

Any HTTP request to the Domino server will cause log records to be written to access and error logs. These logs can be put in text files, a Notes database, or both. The logging fields in the server document HTTP section (Access Log and Error Log) direct the log entries to text files. The presence of domlog.nsf directs log entries to the database.

Logging in Domino is very intensive. You should consider this when deciding to create a log database. For instance, entering http://servername?Open from a Web browser can create four entries in the domlog.nsf database.

5.4.1 Domino Log Database

If you decide to send the log records to a database you must create the domlog.nsf database using domlog.ntf. The basic design of the log database provides one form for log records and one view to display them. There is also an agent that runs on a daily schedule that deletes documents older than ten days. You may want to modify this if you find ten days worth of information is too much or too little. To get the agent to work you must go in and change the agent daily schedule to run on your server.

The Requests view shows all records in the order they were created. Figure 86 on page 93 is an example of what this view will look like.

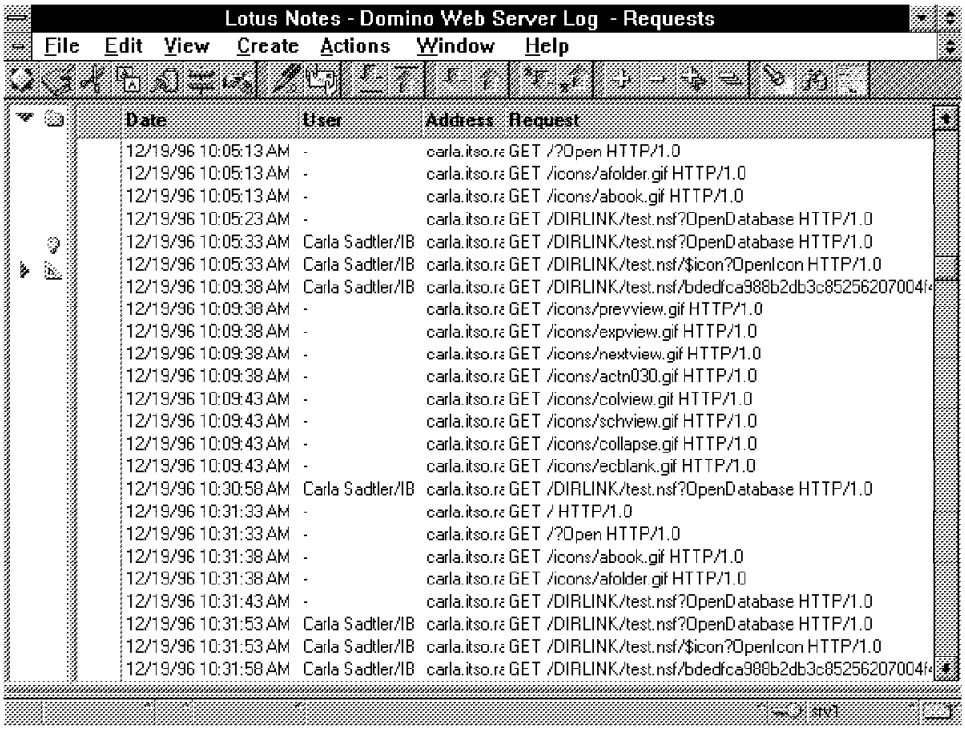


Figure 86. Domino Log Database Request View

The LogEntry form is the form used for all logged events. Figure 87 shows an example log record.

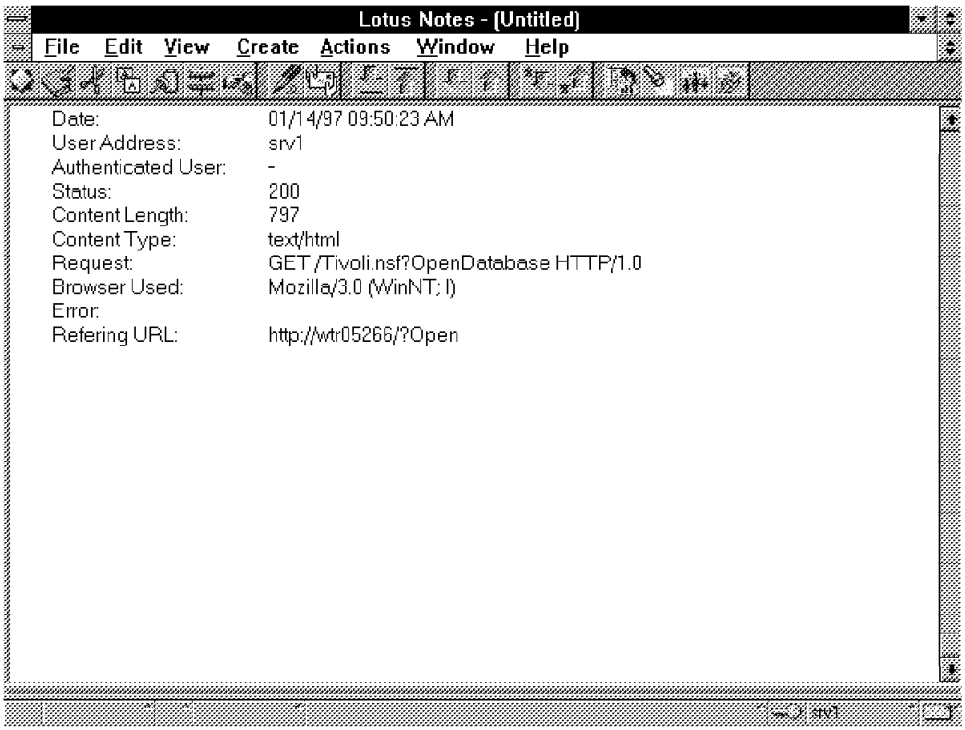


Figure 87. Domino Log Database Entry

5.4.1.1 Tailoring the Domino Log Database

To make this database more useful to us we did some tailoring to the design. We created several views we found to be useful and added agents to automate events we wanted to be aware of.

To allow easy monitoring of invalid login attempts we created a view showing only those records. To select the records we chose those that have a username filled in but a 401 status code. A record with a 401 status code and no name would indicate the initial attempt to access a protected object. The user would then have the opportunity to log in. If this is unsuccessful, the record with the user's name and the 401 status code is built. The view would appear as in Figure 88. The second column is for counting the login attempts. It has a value of one and is totaled for each user and for the entire column.

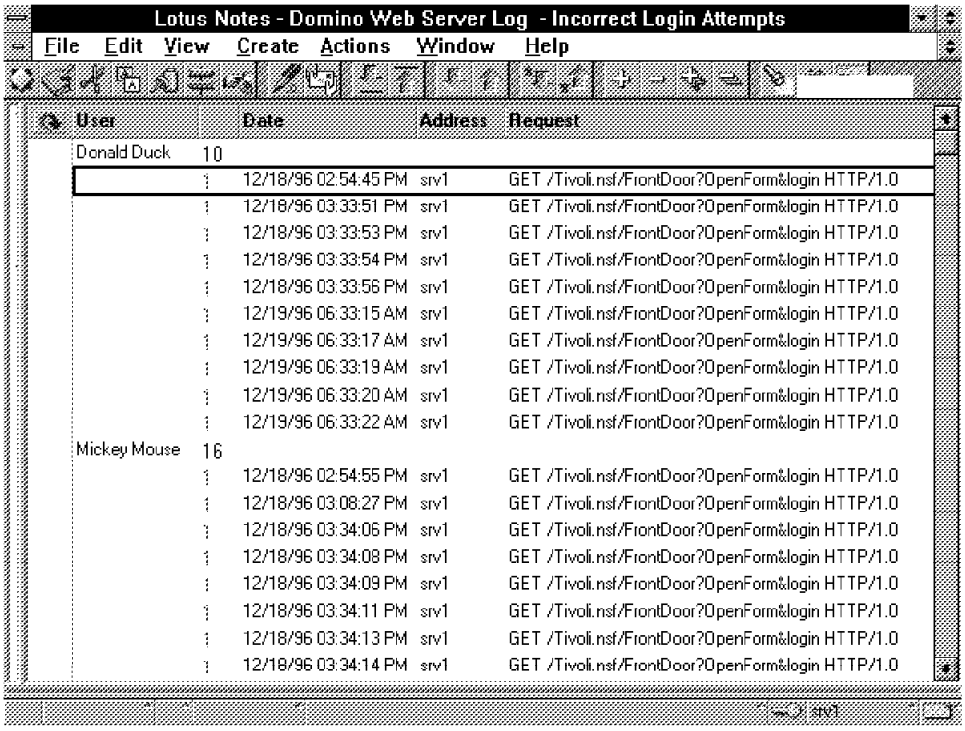


Figure 88. Domino Log Database Incorrect Login Attempts View

The design of our view is shown in Figure 89 on page 95.

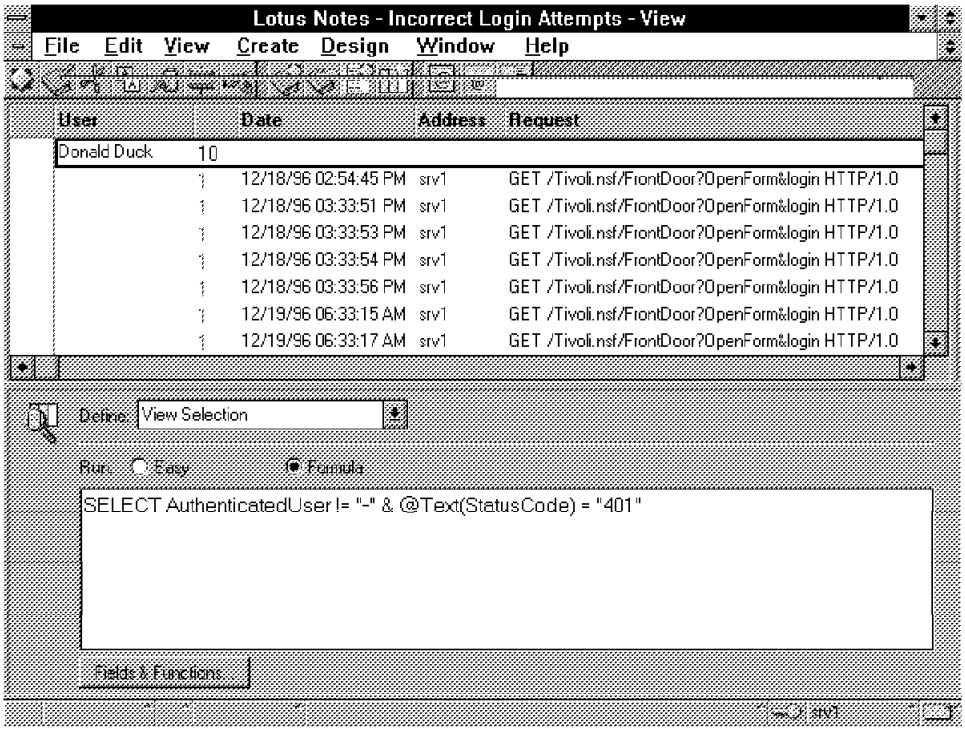


Figure 89. Domino Log Database Incorrect Login Attempts View Design

Other views can be created to allow easy viewing of particular records. Selecting records based on status codes can be very useful. The status codes found recorded in the LogEntry documents are an important descriptor of the event that was recorded. A brief meaning of these codes is listed in the following sections.

5.4.1.2 2xx Status Codes - Success

200 Request fulfilled.

201 Document created.

202 Request accepted but processing not complete.

203 Partial information received.

204 No response.

5.4.1.3 3xx Status Codes - Redirection

301 Data has moved.

302 Data has moved.

303 Data has been found under a different URL and a different method (rather than GET) may be needed.

304 A conditional GET has been done but the document has not been modified since the date and time specified in the If-Modified-Since field.

5.4.1.4 4xx Status Codes - Client Error

400 Bad request.

401 Unauthorized request.

402 Payment required.

403 Forbidden request. (Authorization will not help.)

403 URL not found.

5.4.1.5 5xx Status Codes - Server Error

500 Internal error.

501 Facility required is not implemented.

502 Server temporarily overloaded and can't process request.

503 Gateway timeout.

5.4.1.6 Adding Agents to the Domino Log Database

Browsing log files is not a very exciting prospect, so it would be useful to have tools to extract interesting events from the file. One advantage of using the Domino Log database is that it allows the administrator to use Notes features to do this. Creating agents to check for specific events is a good way to audit what is going on in the Domino server.

As a simple example, we created an agent that runs on an hourly basis on the Domino server. The agent will scan through the Domino Log database using the Incorrect Login Attempts view to check for a large number of unsuccessful login attempts. If the agent finds that a user has tried to log in unsuccessfully more than ten times, a note is sent to the Notes administrator.

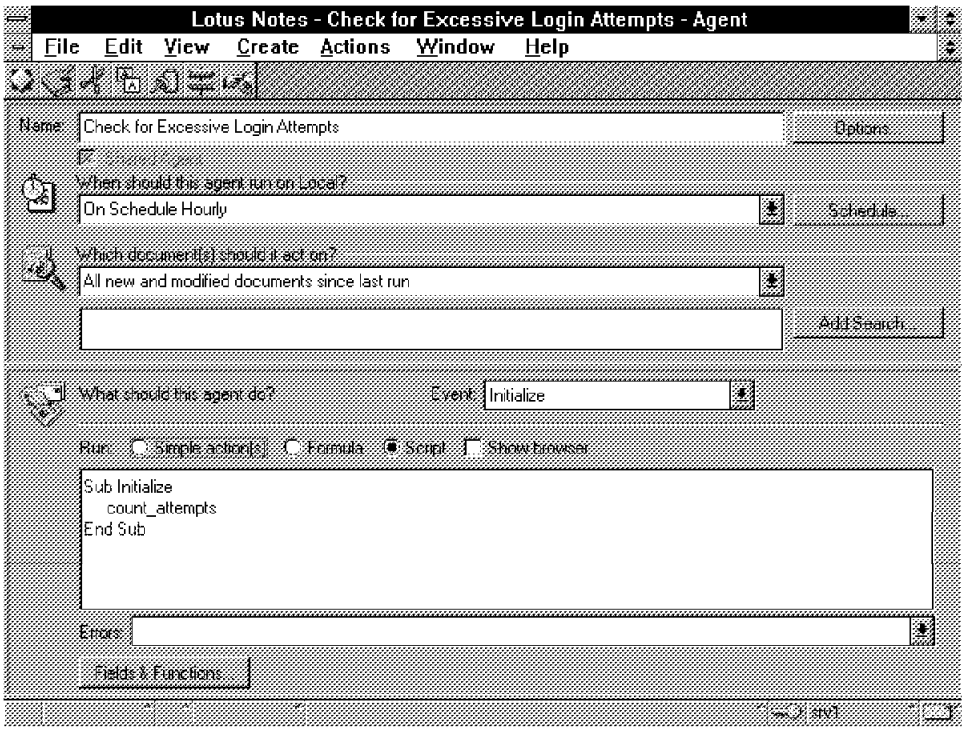


Figure 90. Domino Log Database Agent to Check Excessive Login Attempts

The agent script initialization calls for a subroutine named count_attempts. The code for that subroutine is shown in Figure 91 on page 97.

```
Sub count_attempts
  Dim session As New NotesSession
  Dim db As New NotesDatabase ("srv1","domlog.nsf")
  Dim view As NotesView
  Set View = db.getview("Incorrect Login Attempts")
  Dim doc As NotesDocument
  Dim doc2 As New NotesDocument (db)
  Dim mlist ()
  Redim mlist (4)
  Dim NumberOfTries As Integer

  Set doc = view.getfirstdocument
  If Not (Doc Is Nothing) Then
       ThisUser = doc.getitemvalue("AuthenticatedUser")
  Else
       ThisUser(0) = " "
  End If

  While Not ( Doc Is Nothing)
    Hacker = doc.getitemvalue("AuthenticatedUser")
    NumberOfTries = 0

    While ThisUser(0) = Hacker(0)
      NumberOfTries = NumberOfTries + 1
      Set doc = view.getnextdocument(doc)
      If Not (Doc Is Nothing) Then
           ThisUser = doc.getitemvalue("AuthenticatedUser")
      Else
           ThisUser(0) = " "
      End If
    Wend
    If NumberOfTries > 10 Then
      Set doc2 = New NotesDocument(db)
      doc2.form = "Memo"
      mlist (0) = "Carla Sadtler/IBMITSO"
      mlist (1) = "Rob Mcgregor/IBMITSO"
      doc2.subject = "RE: "+Hacker(0)+" has attempted "+Cstr(NumberOfTries) + " logins"
      doc2.body = "Please check the domino log to see if this is a hacker !"
      Call doc2.send(True,mlist)
    End If
  Wend
End Sub
```

Figure 91. Agent Script Code for Count_Attempts Subroutine

5.4.2 Domino Log Text Files

If you choose to have log records sent to text files you must fill in the Access Log and Error Log fields in the HTTP section of the server document. A new text file is created for each log every day. The files created will have the date appended to their name.

If the server document HTTP section has a value filled in the Access Log field, three files will be created. The access log using the prefix specified in the Access Log field plus the date, the agent log named agent_log.date, and the referer log named referer_log.date.

If the server document HTTP section has a value filled in the Error Log field, two files will be created. The error log using the prefix specified in the Error Log field plus the date and the CGI error log named cgi_error.date.

An entry is created in the access log every time the server receives a request. Figure 92 on page 98 shows an example of an access log.

```
mcgregor.itso.ral.ibm.com - - [07/Nov/1996:11:53:27 +0000] "GET / HTTP/1.0" 200 868
mcgregor.itso.ral.ibm.com - - [07/Nov/1996:11:53:27 +0000] "GET /frntdoor.nsf/90e910a2c3c08faf852563d8005925c9/$ViewMapLayout/0.248?OpenElem" 200 2959
mcgregor.itso.ral.ibm.com - - [07/Nov/1996:11:55:44 +0000] "GET /frntdoor.nsf/FrontDoor?OpenForm HTTP/1.0" 401 262
mcgregor.itso.ral.ibm.com - - [07/Nov/1996:11:55:55 +0000] "GET /frntdoor.nsf/FrontDoor?OpenForm HTTP/1.0" 401 262
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:15:28 +0000] "GET / HTTP/1.0" 200 868
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:15:28 +0000] "GET /frntdoor.nsf/90e910a2c3c08faf852563d8005925c9/$ViewMapLayout/0.248?OpenElem" 200 2959
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:15:38 +0000] "GET /test1.nsf?OpenDatabase HTTP/1.0" 200 591
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:15:38 +0000] "GET /test1.nsf/$icon?OpenIcon HTTP/1.0" 200 1154
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:15:39 +0000] "GET /test1.nsf/dcfa147c29467635852563c700554f28?OpenView HTTP/1.0" 200 406
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:15:47 +0000] "GET /frntdoor.nsf/FrontDoor?OpenForm HTTP/1.0" 401 262
wtr05266.itso.ral.ibm.com - Web User1 [07/Nov/1996:16:15:55 +0000] "GET /frntdoor.nsf/FrontDoor?OpenForm HTTP/1.0" 200 961
wtr05266.itso.ral.ibm.com - Web User1 [07/Nov/1996:16:16:01 +0000] "POST /frntdoor.nsf/9a2efe4e6540ce5f852563d5005aaee0?CreateDocument HTTP/1.0" 302 394
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:16:01 +0000] "GET /test1.nsf?OpenDatabase HTTP/1.0" 200 591
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:16:02 +0000] "GET /test1.nsf/$icon?OpenIcon HTTP/1.0" 200 1154
wtr05266.itso.ral.ibm.com - - [07/Nov/1996:16:16:03 +0000] "GET /test1.nsf/dcfa147c29467635852563c700554f28?OpenView HTTP/1.0" 200 406
wtr05266.itso.ral.ibm.com - Web User1 [07/Nov/1996:16:16:05 +0000] "GET /frntdoor.nsf/FrontDoor?OpenForm HTTP/1.0" 200 961
```

Figure 92. Domino Access Log Example (notes\data\access_log.Nov0796)

The agent log keeps a record of the Web client types that access the Domino server. Figure 93 shows an example of an agent log.

```
[06/Nov/1996:13:31:39 +0000] "Mozilla/3.0 (WinNT; I)"
[06/Nov/1996:13:34:19 +0000] "IBM WebExplorer /v1.01"
[06/Nov/1996:13:35:42 +0000] "Mozilla/3.0 (X11; I; AIX 2)"
[06/Nov/1996:13:35:42 +0000] "Mozilla/3.0 (X11; I; AIX 2)"
```

Figure 93. Domino Agent Log Example (notes\data\agent_log.Nov0796)

The error log records internal errors. Figure 94 shows an example error log.

```
[07/Nov/1996:11:08:28 +0000] [OK] [host: wtr05266.itso.ral.ibm.com referer: http://wtr05266.itso.ral.ibm.com/frntdoor.nsf?OpenDatabase] /frntdoor.nsf/FrontDoor
[07/Nov/1996:11:10:40 +0000] [OK] [host: 0.0.0.0] Request parsing failed fatally
[07/Nov/1996:11:10:40 +0000] [OK] [host: 0.0.0.0] Request parsing failed fatally
[07/Nov/1996:11:53:56 +0000] SSL Handshake failed
[07/Nov/1996:15:10:35 +0000] [OK] [host: 0.0.0.0] Request parsing failed fatally
[07/Nov/1996:16:15:47 +0000] [OK] [host: wtr05266.itso.ral.ibm.com referer: http://wtr05266.itso.ral.ibm.com/] /frntdoor.nsf/FrontDoor
[07/Nov/1996:16:16:32 +0000] [OK] [host: wtr05266.itso.ral.ibm.com referer: http://wtr05266.itso.ral.ibm.com/?Open] /DIRLINK/test.nsf
[07/Nov/1996:16:19:05 +0000] [OK] [host: wtr05266.itso.ral.ibm.com] /
```

Figure 94. Domino Error Log Example (notes\data\eror_log.Nov0796)

The CGI error log has records of standard error (stderr) data from CGI programs. The error log will be in the same path you specify for the error log and will be named cgi_error.date.

The referer log has records indicating sites that contain links to this site. Figure 95 shows an example referer log.

```
[07/Nov/1996:11:53:27 +0000] "http://wtr05266/"
[07/Nov/1996:11:55:44 +0000] ""
[07/Nov/1996:11:55:55 +0000] ""
[07/Nov/1996:16:15:28 +0000] ""
[07/Nov/1996:16:15:28 +0000] "http://wtr05266.itso.ral.ibm.com/"
[07/Nov/1996:16:15:38 +0000] "http://wtr05266.itso.ral.ibm.com/"
[07/Nov/1996:16:15:38 +0000] "http://wtr05266.itso.ral.ibm.com/test1.nsf?OpenDatabase"
[07/Nov/1996:16:15:39 +0000] "http://wtr05266.itso.ral.ibm.com/test1.nsf?OpenDatabase"
[07/Nov/1996:16:15:47 +0000] "http://wtr05266.itso.ral.ibm.com/"
[07/Nov/1996:16:15:55 +0000] "http://wtr05266.itso.ral.ibm.com/"
[07/Nov/1996:16:16:01 +0000] "https://wtr05266.itso.ral.ibm.com/frntdoor.nsf/FrontDoor?OpenForm"
[07/Nov/1996:16:16:01 +0000] "https://wtr05266.itso.ral.ibm.com/frntdoor.nsf/FrontDoor?OpenForm"
[07/Nov/1996:16:16:02 +0000] "https://wtr05266.itso.ral.ibm.com/test1.nsf?OpenDatabase"
[07/Nov/1996:16:16:03 +0000] "https://wtr05266.itso.ral.ibm.com/test1.nsf?OpenDatabase"
[07/Nov/1996:16:16:05 +0000] "http://wtr05266.itso.ral.ibm.com/"
[07/Nov/1996:16:16:19 +0000] ""
```

Figure 95. Domino Referer Log Example (notes\data\referer_log.Nov0796)

5.4.2.1 Statistics

The SHOW STAT DOMINO console command can be used to find statistics about the Domino server. Entering the command will show results that look like those in Figure 96.

```
> show stat domino
  Domino.BuildName = Beta 3
  Domino.BuildNumber = 145
  Domino.BuildPlatform = NT/Intel
  Domino.BuildVersion = Lotus Domino Web Server Release 1.5
  Domino.Cache.Command.DisplaceRate = 0
  Domino.Cache.Command.HitRate = 0
  Domino.Cache.Database.DisplaceRate = 0
  Domino.Cache.Database.HitRate = 80.8219178082192
  Domino.Command.CreateDocument = 1
  Domino.Command.DeleteDocument = 0
  Domino.Command.EditDocument = 0
  Domino.Command.Login = 0
  Domino.Command.Navigate = 0
  Domino.Command.OpenAbout = 0
  Domino.Command.OpenAgent = 0
  Domino.Command.OpenDatabase = 17
  Domino.Command.OpenDocument = 6
  Domino.Command.OpenElement = 7
  Domino.Command.OpenForm = 10
  Domino.Command.OpenHelp = 0
  Domino.Command.OpenIcon = 6
  Domino.Command.OpenNavigator = 18
  Domino.Command.OpenServer = 2
  Domino.Command.OpenView = 8
  Domino.Command.ReadForm = 0
  Domino.Command.SaveDocument = 0
  Domino.Command.SearchSite = 0
  Domino.Command.SearchView = 0
  Domino.Command.Total = 75
  Domino.Command.Unknown = 0
  Domino.Config.ActiveThreads.Max = 40
  Domino.Config.ActiveThreads.Min = 20
  Domino.Config.AllowDirectoryLinks = 0
  Domino.Config.Cache.Delete = 0
  Domino.Config.Cache.Directory = d:\notes\data\domino\cache
  Domino.Config.Cache.Size.Max = 50
  Domino.Config.Directory.CGI = d:\notes\data\domino\cgi-bin
  Domino.Config.Directory.HTML = d:\notes\data\domino\html
  Domino.Config.Directory.Icons = d:\notes\data\domino\icons
  Domino.Config.DNSLookup = 1
  Domino.Config.GarbageCollection.Interval = 60
  Domino.Config.GarbageCollection.Status = 1
  Domino.Config.HomeURL = /frntdoor.nsf?OpenDatabase
  Domino.Config.HostName = wtr05266.itso.ral.ibm.com
  Domino.Config.Image.Format = GIF
  Domino.Config.Image.Interlaced = 1
  Domino.Config.Log.Access =
  Domino.Config.Log.Error =
  Domino.Config.Log.Filter =
  Domino.Config.Log.TimeStamp = 0
  Domino.Config.PortNumber = 80
  Domino.Config.PortStatus = 1
  Domino.Config.SSL.KeyFile = v1s512.kyr
  Domino.Config.SSL.PortNumber = 443
  Domino.Config.SSL.Status = 1
  Domino.Config.Timeout.CGI = 5
  Domino.Config.Timeout.IdleThread = 0
  Domino.Config.Timeout.Input = 2
  Domino.Config.Timeout.Output = 20
  Domino.Config.URLpath.CGI = /cgi-bin
  Domino.Config.URLpath.Icons = /icons
  Domino.Config.View.Lines = 30
  Domino.Config.WelcomePage = default.htm
  Domino.Requests.Per1Day.Peak = 58
  Domino.Requests.Per1Day.PeakTime = 11/05/96 18:23:04
  Domino.Requests.Per1Day.Total = 58
  Domino.Requests.Per1Hour.Peak = 77
  Domino.Requests.Per1Hour.PeakTime = 11/05/96 18:24:10
  Domino.Requests.Per1Hour.Total = 77
  Domino.Requests.Per1Minute.Peak = 34
  Domino.Requests.Per1Minute.PeakTime = 11/05/96 18:21:54
  Domino.Requests.Per1Minute.Total = 12
  Domino.Requests.Per5Minute.Peak = 69
  Domino.Requests.Per5Minute.PeakTime = 11/05/96 18:24:10
  Domino.Requests.Per5Minute.Total = 68
  Domino.Requests.Total = 77
  Domino.StartTime = 11/05/96 18:17:57
  Domino.Threads.Active.Peak = 1
  Domino.Threads.Peak.Time = 11/05/96 18:18:03
  Domino.Threads.Peak.Total = 20
  Domino.Threads.Total = 20
  Domino.zThread.On_00 = 99
  Domino.zThread.On_01 = 99
```

Figure 96. Domino SHOW STAT DOMINO Command

5.4.2.2 Notes System Log (LOG.NSF)

In the NOTES.INI configuration file, the LOG_SESSIONS parameter should be 1 in order to log individual sessions. If you do this, the server logs errors related to HTTP sessions, for example `You are not authorized to perform that operation`, in the same manner as that for Notes user.

```
[NOTES]
LOG_SESSIONS=1
```

Figure 97. NOTES.INI Setting to Log Domino Error Messages

The following is the fragment of Notes log from LOG.NSF database.

```
11/07/96 05:15:47 PM  HTTP Web Server: Lotus Notes Exception - You are not authorized to perform that operation
                      [/frntdoor.nsf/FrontDoor?OpenForm]
11/07/96 05:16:56 PM  HTTP Web Server: Lotus Notes Exception - You are not authorized to perform that operation
                      [/mail/wjasmin.nsf?OpenDatabase]
11/07/96 05:17:32 PM  HTTP Web Server: Lotus Notes Exception - File is not a database [/DIRLINK/test.nsf?OpenDatabase]
11/07/96 05:19:05 PM  HTTP Web Server: Database Browsing Not Allowed [/?Open]
```

5.4.2.3 Database Activity Record

You must record activities for each database. To do this, open database property, go to Information page, click on **User Activity...**, and check Record Activity.

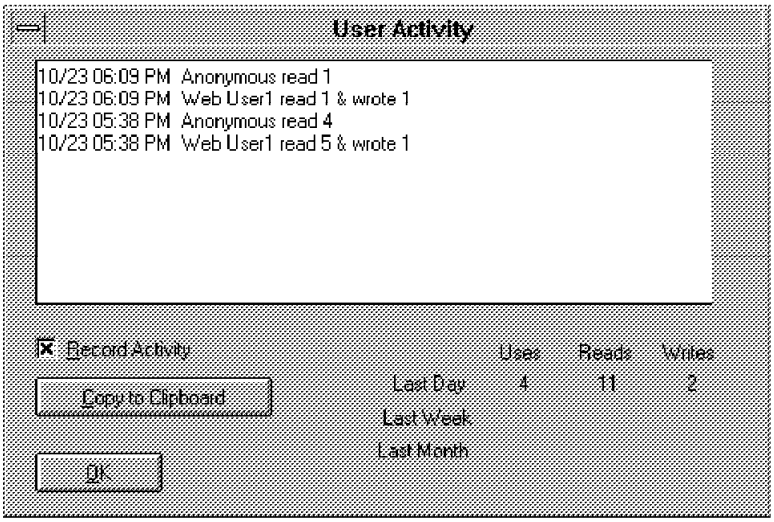


Figure 98. Database Properties - Information - User Activity

Chapter 6. An Example Application

In this chapter we describe a sample application that illustrates how many of the Notes security capabilities are applied, and how the resulting application can be extended into the Internet using Domino. The application is a prototype database designed to track questions and answers about IBM′s systems management offering, the Tivoli Management Environment.

The database contains two parts, a Q&A database which is designed to hold question and answer information and a discussion forum where users can discuss Tivoli subjects.

The registration application and the Frontdoor applications covered in earlier chapters have been incorporated into this example. The Frontdoor sample has been modified for our purposes and merged with the main database.

The Domino Registration sample (available for download at http://domino.lotus.com) remains separate. We have changed the group name the new user is put in to submitter_group in the GroupsToJoin shared field.

Users entering the Tivoli database have a choice of immediately logging on and using SSL if needed for their session, or using unsecure transmission. A user choosing unsecure transmission will be asked to log on in the application the first time they attempt to access a restricted portion of the database. Until then the user ID in use is Anonymous.

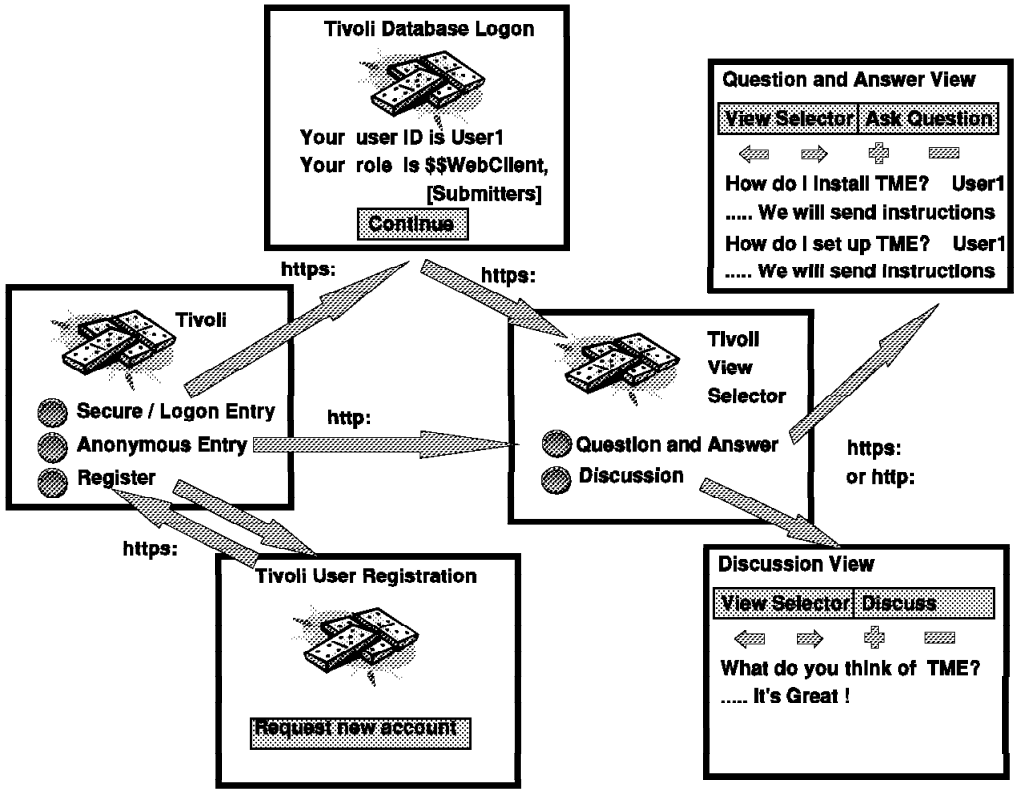


Figure 99. Tivoli Database Flow

© Copyright IBM Corp. 1997

6.1 Tivoli Database Users

There are four types of users that can access the database:

- **Administrators** are ITSO staff members who have the authority to read or create any document. They are the only users that may enter answers into the database. Because they can read and create restricted answers they should enter the database through the Secure Entry button.
- **Trusted Users** are users who have clearance to see restricted material on Tivoli subjects. They may enter questions and discussion entries. They are allowed to read all entries, including restricted answers. Because they can read restricted answers they should enter the database through the Secure Entry button.
- **Submitters** are general users who are registered to the database and may enter questions and discussion entries. They may read unrestricted answers, all discussion entries, and all questions.
- **Subscribers** are Anonymous users. They are allowed to read discussion entries, but no questions or answers.

6.2 Tivoli Database Access Control

The access control is organized with Public Address Book groups and database roles.

Users are in one of four groups in the Public Name and Address Book:

- admin_group
- trustuser_group
- submitter_group
- subscriber_group

They are entered manually or they can register through the Domino Registration sample database. Users who register this way are automatically put in submitter_group. Being a member of subscriber_group and being anonymous give you the same authorities. Administrators and trusted users must be entered manually into admin_group and trustuser_group respectively. These groups are assigned roles in the database.

We chose to use roles for simplicity. We used the @UserRoles function in several places in the design to prevent users from seeing views they should not see and to hide actions based on a user's authority.

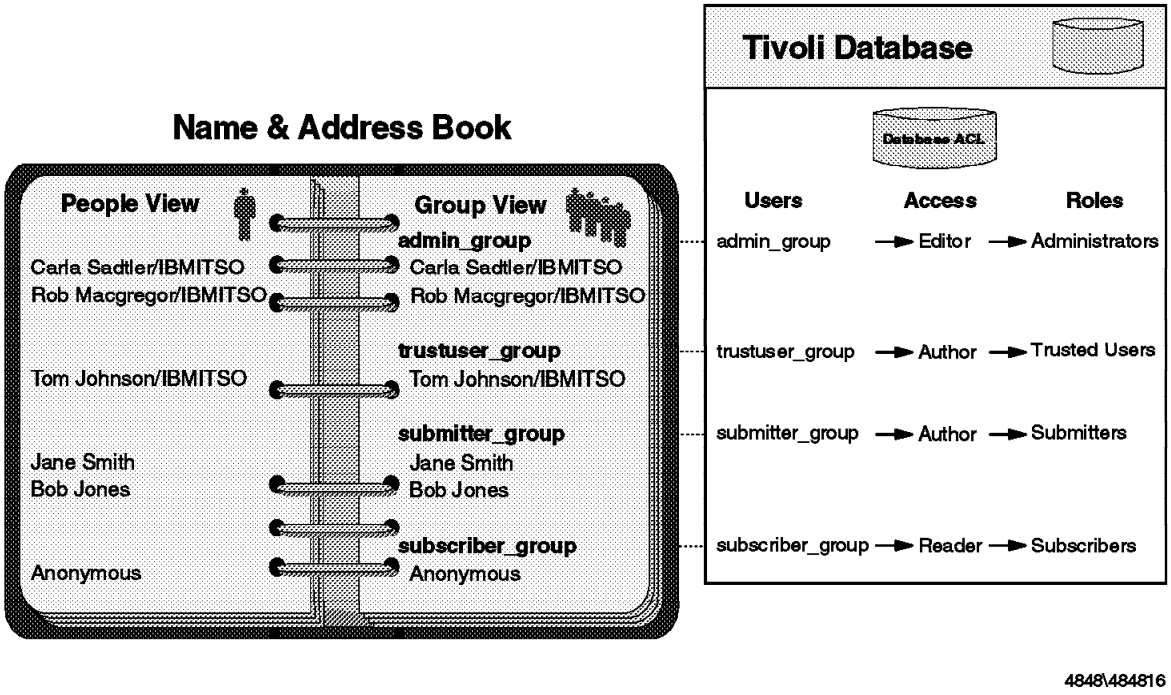


Figure 100. Tivoli Database Access Control

6.3 Initial Database Entry

The Tivoli database is designed to launch an initial navigator. This navigator, seen Figure 101 on page 104, allows the user to choose one of three options:

- The user may choose the Secure/Logon Entry button to log on to the database and use SSL.
- The user may enter the database as an Anonymous user by using unsecured transmission by clicking the Anonymous Entry button.
- An unregistered user can register as a subscriber of the database.

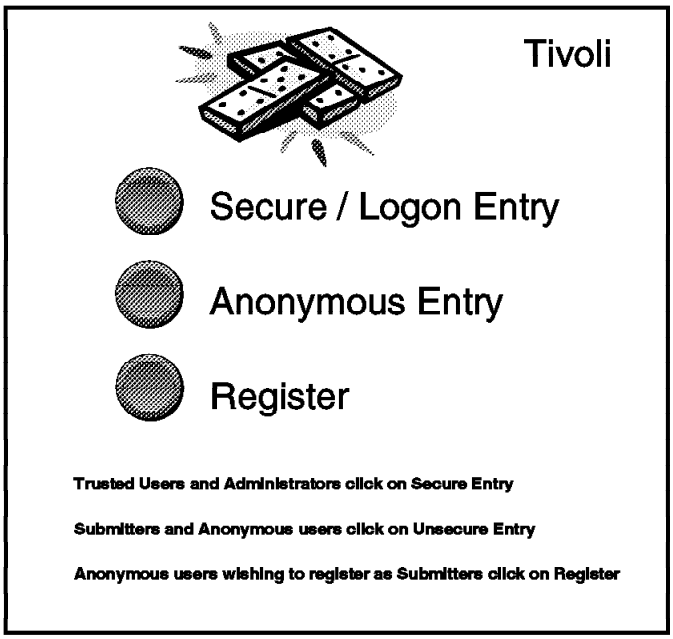


Figure 101. Tivoli Database Welcome Page

6.3.1 Initial Navigator Design

The initial navigator was created by using a background graphic with hotspot rectangles located on each button. The hotspots use a formula or simple action to navigate to the next location. For actions that require SSL, we built a formula consisting of a URL with https:// to navigate to the next object. For unsecure actions we use the simple actions provided when creating a hotspot rectangle. The simple action will be translated by Domino to use the mode we entered the navigator in, in this case non-SSL using http//.

Table 5. Initial Navigator Design

Hotspot	Formula or Simple Action
Secure/Logon Entry	@URLOpen(″https://srv1/Tivoli.nsf/FrontDoor?OpenForm&login″)
Anonymous Entry	Simple Action - Open another navigator - View Selector
Register	@URLOpen(″https://srv1/DOMREG.NSF?OpenDatabase″)

6.4 Secure Entry

Administrators and trusted users should enter the database by selecting the Secure/Logon Entry button. Since they have access to restricted answers this path will provide security by forcing the user to log on initially and changing the session to use SSL. Submitters may enter through this path to log on but they will not use SSL.

Clicking the **Secure/Logon Entry** button executes an @URLOpen function to open the FrontDoor form. The URL is coded with https://, causing the session to use SSL during the logon. Once the user is logged on, the formula for a hidden field

in the form, called SSL, is computed. This formula checks the group membership of the user to determine whether the user should use SSL based on the level of database access that the user has.

Once it is determined whether or not the user needs SSL, the $$RETURN field defines a URL using http:// or https:// appropriately.

Once SSL has been used, Domino will continue to build URLs with https:// in formulas it converts to HTML. However, if the design of the database uses a hard coded URL that uses http://, the formula will be executed as written. In other words, you could unintentionally lose the SSL function by hard coding URLs in your database design instead of letting Domino do the translation to HTML.

The function of the FrontDoor form is to cause the user to log on to the database. This can be done by protecting the form using the form properties ACL, by protecting a computed field in the form, or by appending the &login argument to the URL command. The fact that an anonymous user is attempting to access a protected form or field will cause Domino to request a user ID and password. The &login argument will also cause the user to have to log on.

In our sample database, users will enter this form as anonymous users. Since Anonymous has reader access (defined in the subscriber_group) we protected the form at the form ACL level.

The FrontDoor form is restricted for read and create access to:

- Administrators
- Trusted users
- Submitters

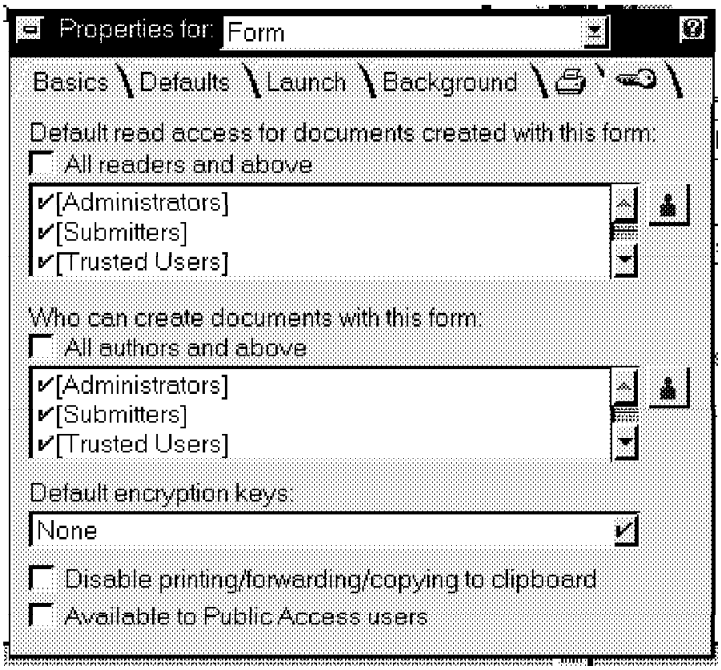


Figure 102. FrontDoor Form Security Properties

Once the user has logged on he or she clicks on the **Continue** button which acts as the submit button. This causes the formula in the $$RETURN field to execute.

The formula for the $$RETURN is defined to send the user to the View Selector navigator. The user is now using SSL and his or her own user ID.

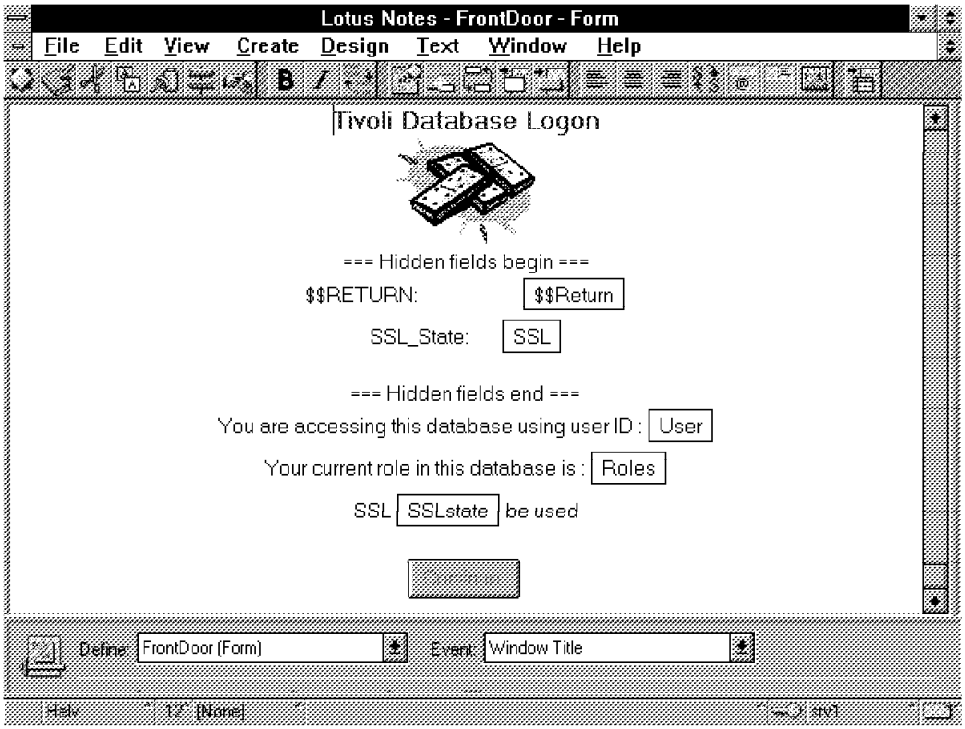


Figure 103. FrontDoor Form Design

Table 6. FrontDoor Form Design

Field	Type	Default Value or Value
$$Return	Text - Computed (hidden)	@If(SSL="1"; "[https://srv1/tivoli.nsf/View+Selector?OpenNavigator]"; "[http://srv1/tivoli.nsf/View+Selector?OpenNavigator]")
SSL	Text - Computed (hidden)	@If(@IsMember("[Administrators]";@UserRoles);"1"; @IsMember("[Trusted Users]";@UserRoles);"1";"0")
User	Text - Computed	@Name([Abbreviate];@UserName)
Roles	Text - Computed Allow multi-values	@UserRoles
SSLstate	Text - Computed	@If(SSL="1";"will";"will not")
SaveOptions	Text - Computed	"0"
Continue button		

Figure 104 on page 107 shows how this form looks in a Web browser when a user who is in the Administrators group logs in. Notice that SSL is active (the key indicator is displayed), because of the high security access of the user.

.

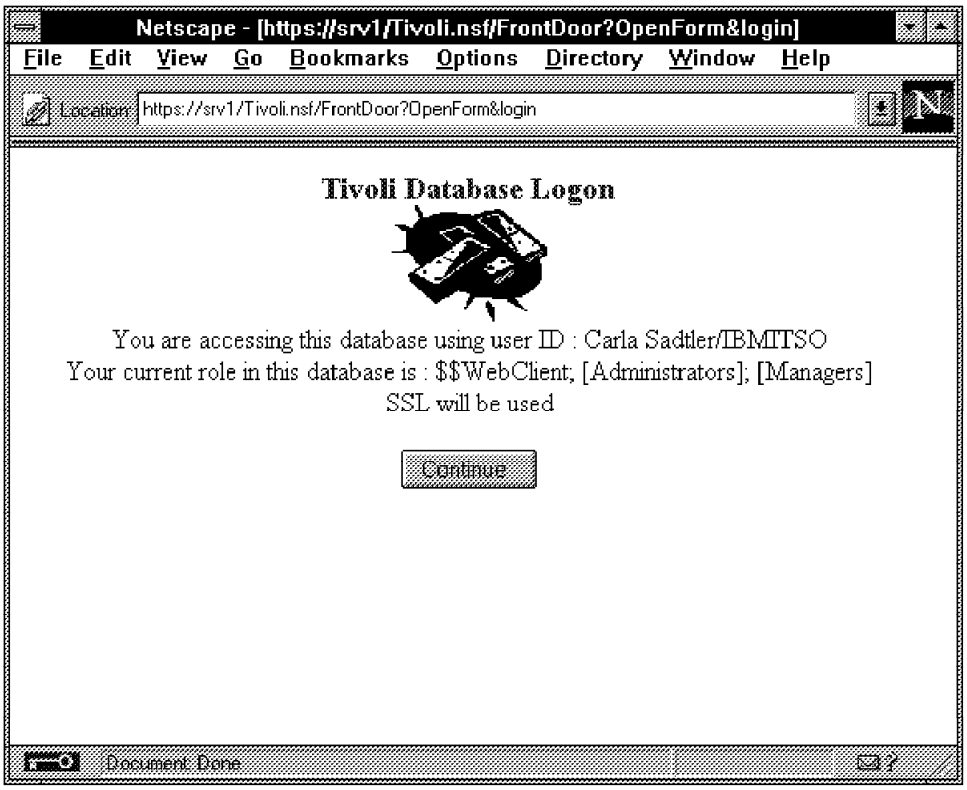


Figure 104. FrontDoor in Web Browser

6.5 View Selector Navigator

When the user selects to enter the database either using secure or unsecure transmission he or she is eventually presented with the View Selector navigator shown in Figure 105 on page 108. If secure transmission was chosen, the navigation is done with URLs built with https://. If unsecure entry was chosen, the URL commands http://.

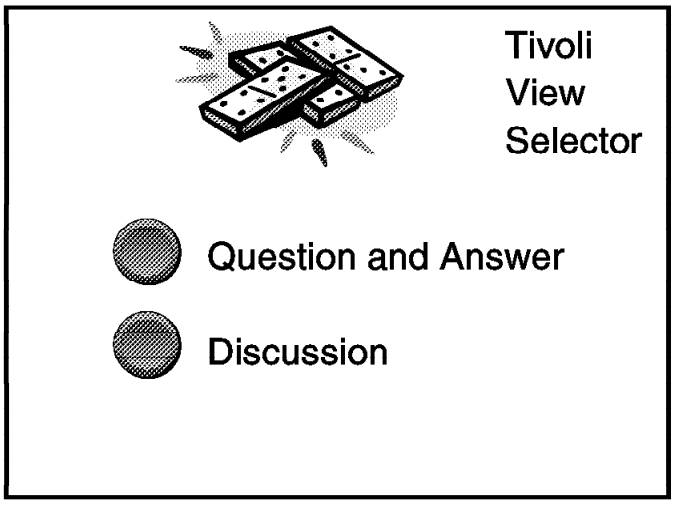


Figure 105. View Selector Navigator

The navigator allows the user to go to the Question and Answer view or the Discussion view. The user is not required to log on to read the discussion entries.

6.5.1 View Selector Navigator Design

The navigator was created by using a background graphic with hotspot rectangles located on each button. For the Discussion hotspot we used the simple actions provided when creating a hotspot rectangle to open the Discussion view.

The Question and Answer view is restricted through the form ACL to the administrators, trusted users, and submitters roles. To avoid getting an error message when unauthorized users try to access this view, we used a formula to keep unauthorized users in the Navigator when the Question and Answer view is chosen. The formula the users′ roles to determine if they are allowed to see the view.

By using the simple actions and @Commands we allow Domino to determine whether to use SSL or not. If the navigator was entered using SSL, Domino continues to use SSL; otherwise, SSL will not be used.

Table 7. View Selector Design

Hotspot	**Formula**
Question and Answer	@If(@Contains(@UserRoles;″[Subscribers]″); @Command([OpenNavigator];″View Selector″); @Command([OpenView];″Question and Answer″))
Discussion	Simple Action - Open a view - Discussion View

6.6 Question and Answer View

The Question and Answer view shows questions, restricted answers and unrestricted answers. It is limited to administrators, submitters, and trusted users by selecting those roles in the view properties security box.

Protecting the Question and Answer view does not cause the user to enter a password, but gives an error on the Web when you go to it from the Navigator. The error can be seen in Figure 106.

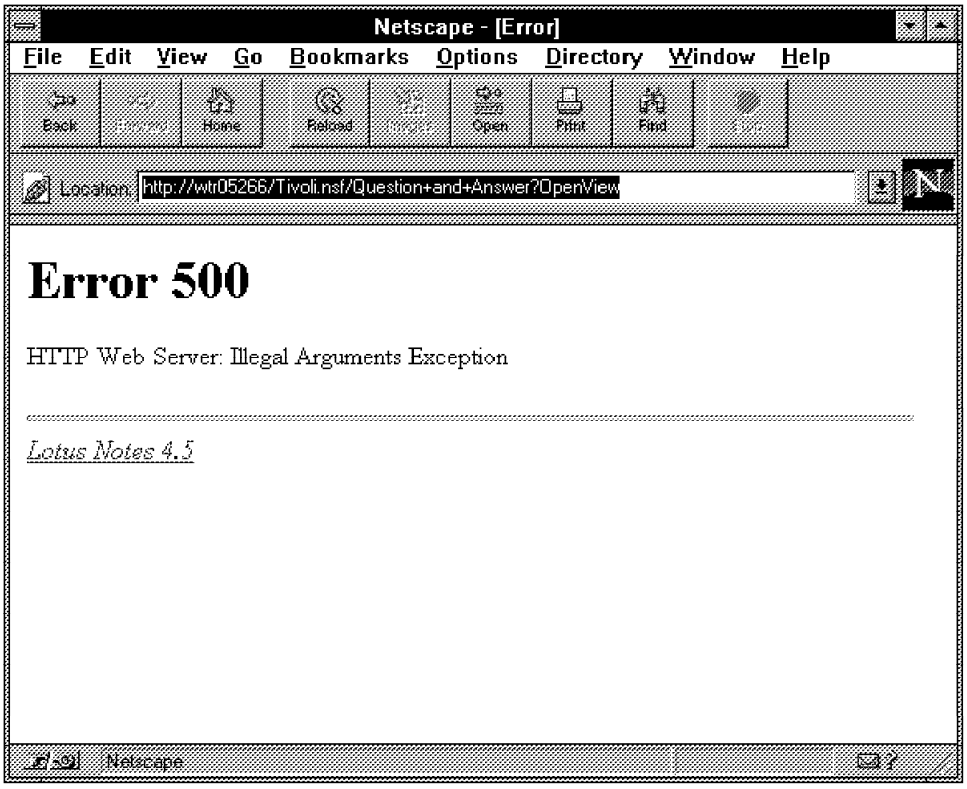


Figure 106. Accessing a Protected View without Logging On

If a user is not allowed access to restricted answers, he or she will still see a twistie beside a question with an answer but will not see an answer entry when he clicks on the twistie.

Did you say twistie?

A *twistie* is the official name for the little triangular icon that sometimes appears beside an entry in a list. It indicates that the entry has another list associated with it. Clicking on the twistie will toggle the display of the subordinate list on and off.

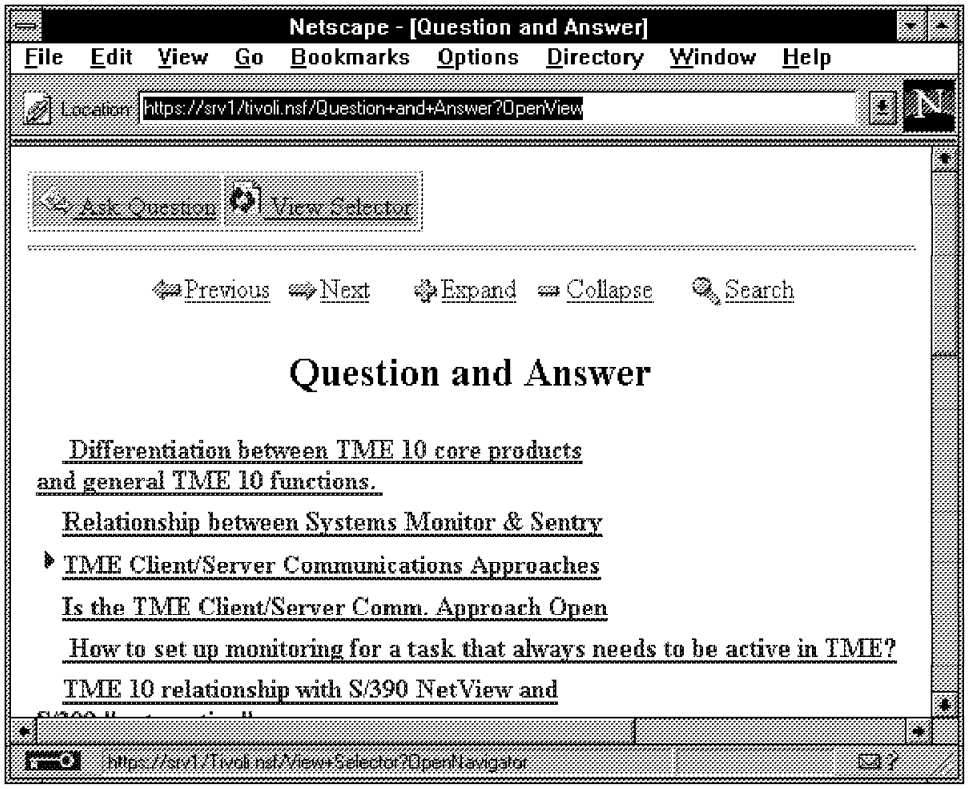


Figure 107. Question and Answer View from Web Browser

6.6.1 Question and Answer View Design

The design of this view is simple. The question and response summaries are shown in a main document / response hierarchy. The questions are the main document and the answers (restricted and unrestricted) are the responses. The first column is the response column. An '*' is added to the response summary if the answer is restricted. We elected not to show the column headings.

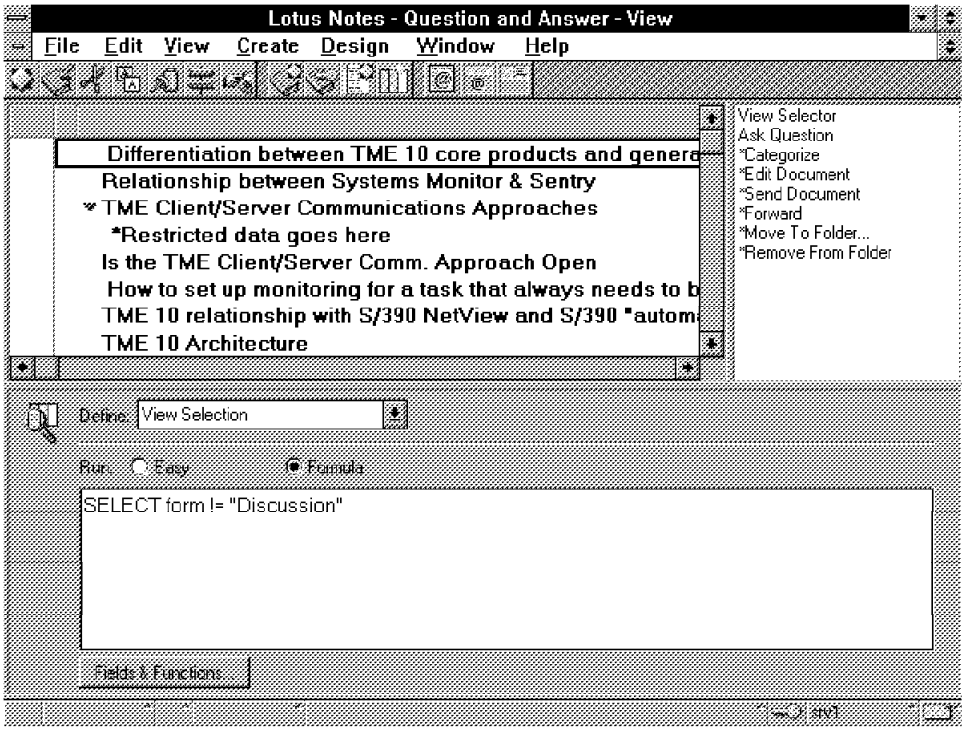


Figure 108. Question and Answer View Design

Table 8. Question and Answer View Design

Column	Basics	Sorting	Formula
1	Width 1 Show Responses Only	None	type:=@if(form="RAnswer";"*";""); type + respsum;
2	Width 47 Show twistie	None	Qsum

The following buttons described in 6.9, “Actions” on page 119 are active on the Question and Answer View:

- View Selector
- Ask Question

6.7 Discussion View

The Discussion view shows all discussion entries, categorized by category and platform. The Discussion form is defined as a document. This means there is no response hierarchy. We use categories and platforms to group the entries.

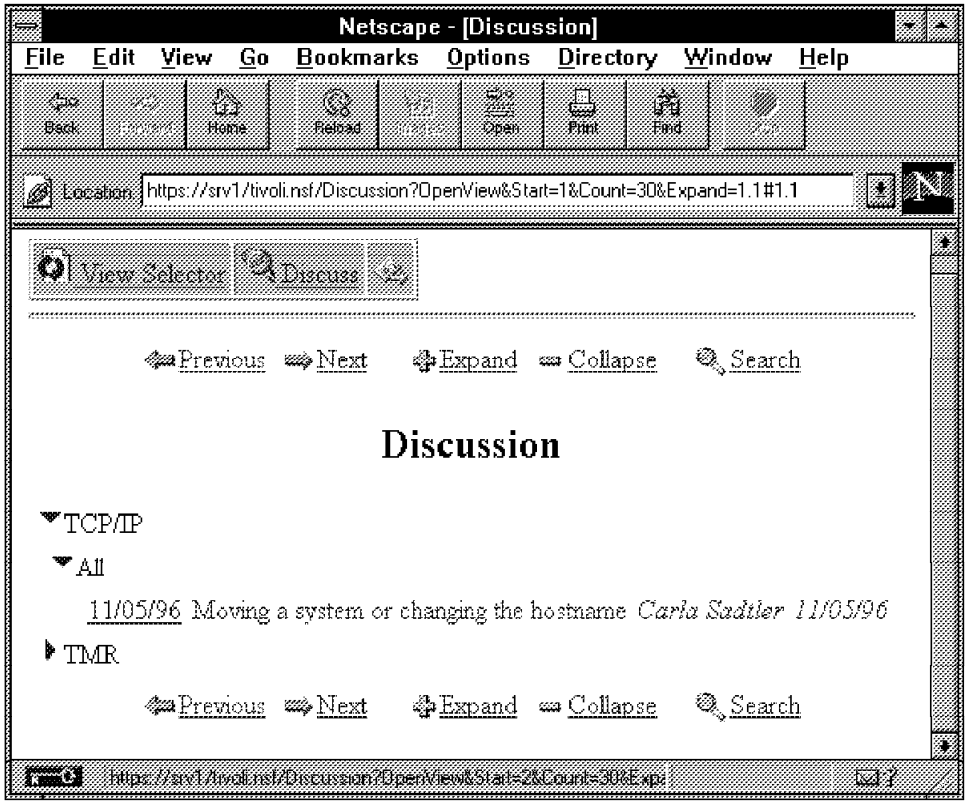


Figure 109. Discussion View from Web Browser

Since discussion entries can be read by anyone with reader access to the database, the Discussion view is accessible to all users with read access or above.

6.7.1 Discussion View Design

We elected not to show the column headers to simplify the way they looked from a Web browser. Four columns are defined. The first two are to categorize the discussion entries by category and platform. The third column is the date the document was created. The fourth shows the summary field of the discussion document.

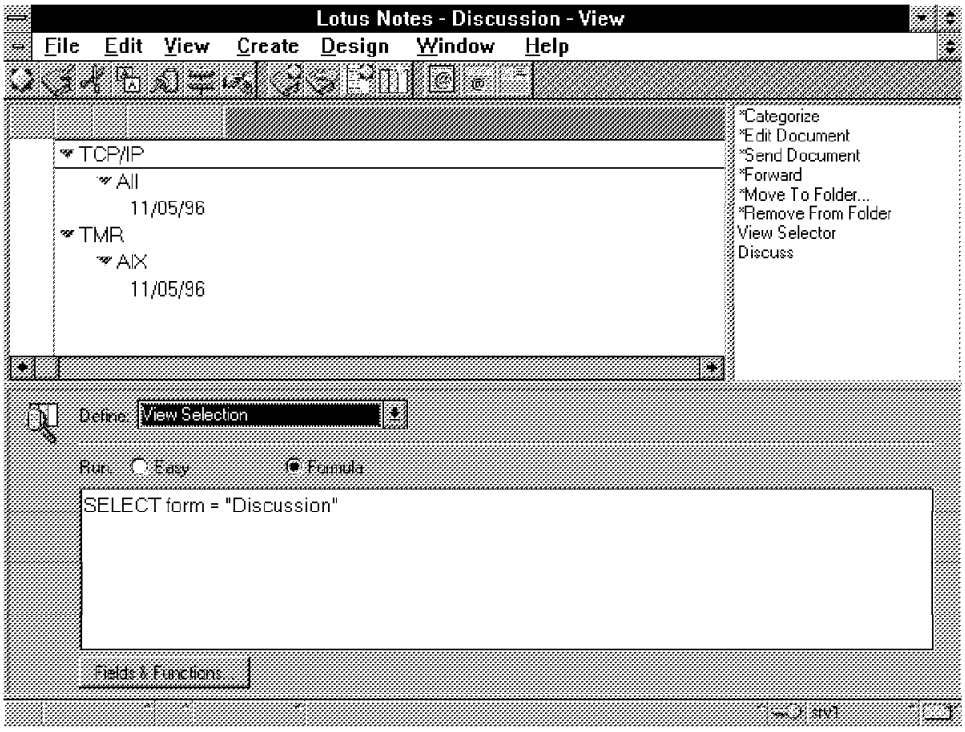


Figure 110. Discussion View Design

Table 9. Discussion View Design

Column	Basics	Sorting	Formula
1	Width 2 Show twistie	Ascending Categorize	Category
2	Width 1 Show twistie	Ascending Categorized	Platform
3	Width 5	Ascending	Date_Created
4	Width 50	None	respsum

The following buttons defined in 6.9, “Actions” on page 119 are active on the Discussion View:

- View Selector
- Discuss

6.8 Forms in the Tivoli Database

There are four forms defined in the database:

- Question
- RAnswer
- UAnswer
- Discussion

Question and Discussion forms are defined as documents. RAnswer (restricted answers) and UAnswer (unrestricted answers) are response documents.

Question and answer forms are threaded together as main documents and responses. Discussion documents are all main documents. They are related by category and platform.

6.8.1 Question Form

Users enter questions into the database using the Question form. The Question form is defined as a document type form with no versioning. To ask a question a user clicks on the **Ask Question** button from the Question and Answer view. Since this view is restricted to the administrators, trusted users, and submitters roles, this button is not seen by those who are not allowed to ask questions.

In addition, the form is protected at the form properties security level to allow read and create access only to administrators, trusted users, and submitters. This prevents unauthorized users from using a URL to open this form directly.

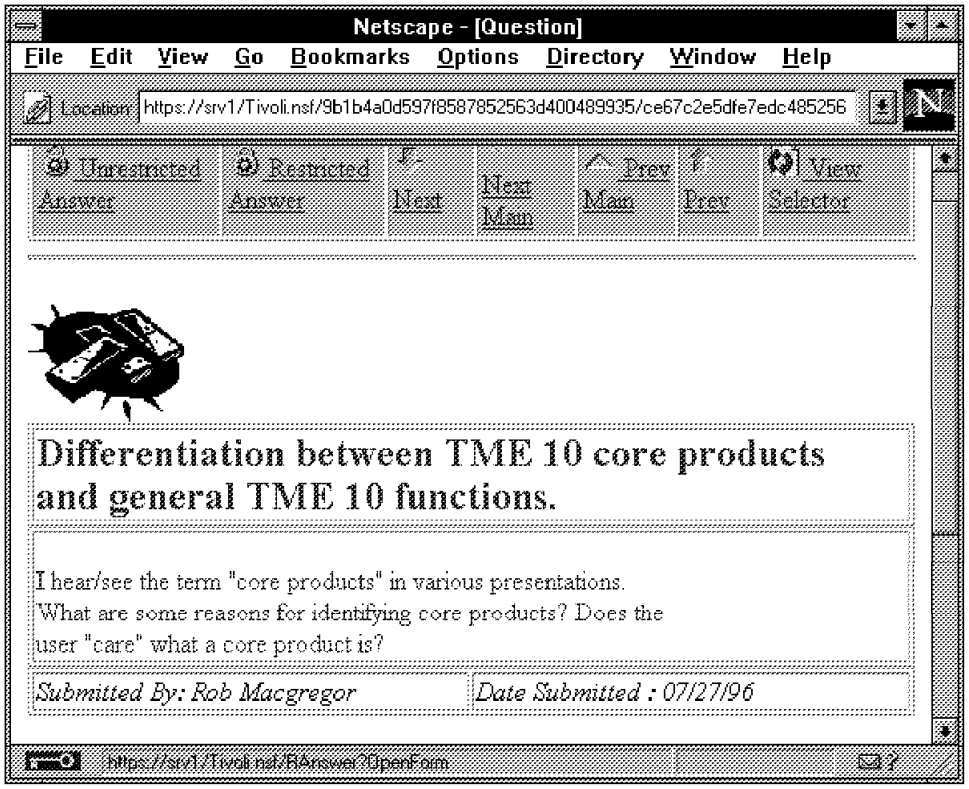


Figure 111. Question Document from Web Browser

The following actions defined in 6.9, “Actions” on page 119 are active:

- Unrestricted Answer
- Restricted Answer
- Next
- Next Main
- Prev Main
- Prev
- View Selector

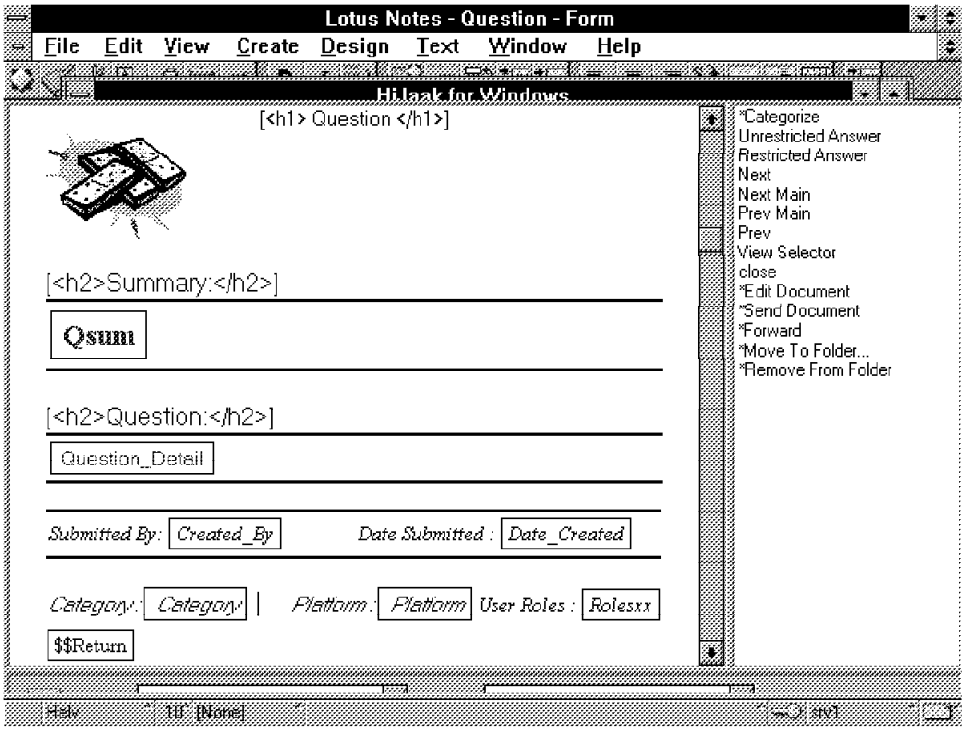


Figure 112. Question Form Design

Table 10. Question Form Design

Field	**Type**	**Default Value or Value**
Qsum	Text - Editable	
Question_Detail	Text - Editable	
Created_By	Computed when composed	@Name([CN];@UserName)
Date_Created	Computed when composed Time (show date)	@Now
Category	Keywords (TME10, etc.), Allow multi-values, Allow values not in list	
Platform	Keywords (AIX,VM,MVS, etc.), Allow multi-values, Allow values not in list	
Rolesxx	Computed for Display	@UserRoles (for user's information only)
$$Return	Computed	@Return(″<h2>Your Question has been submitted </h2> <h4>Return to Question and Answer View″)

6.8.2 Restricted and Unrestricted Answer Forms

The form for restricted answers, RAnswer, and the form for unrestricted answers, UAnswer, look essentially the same to the user, but to simplify security we use different forms. The difference in the two forms is the text at the top and the form security properties (ACL).

Unrestricted answers are protected at the form properties security level to allow only administrators, trusted users, and submitters to read documents with this form. Only administrators are allowed to create documents with this form.

Restricted answers are protected at the form properties security level to allow only administrators and trusted users to read documents with this form. Only administrators are allowed to create documents with this form.

Keep in mind that if you want to change an existing answer document from being unrestricted to restricted it is not enough to change the form. When the document was created it inherited its document read access authority from the form definition. This means once a document is created with the UAnswer form, anyone with administrator, trusted user, or submitter role authority can read it, even if the form name changes. The person changing the form must also change the document property security designations.

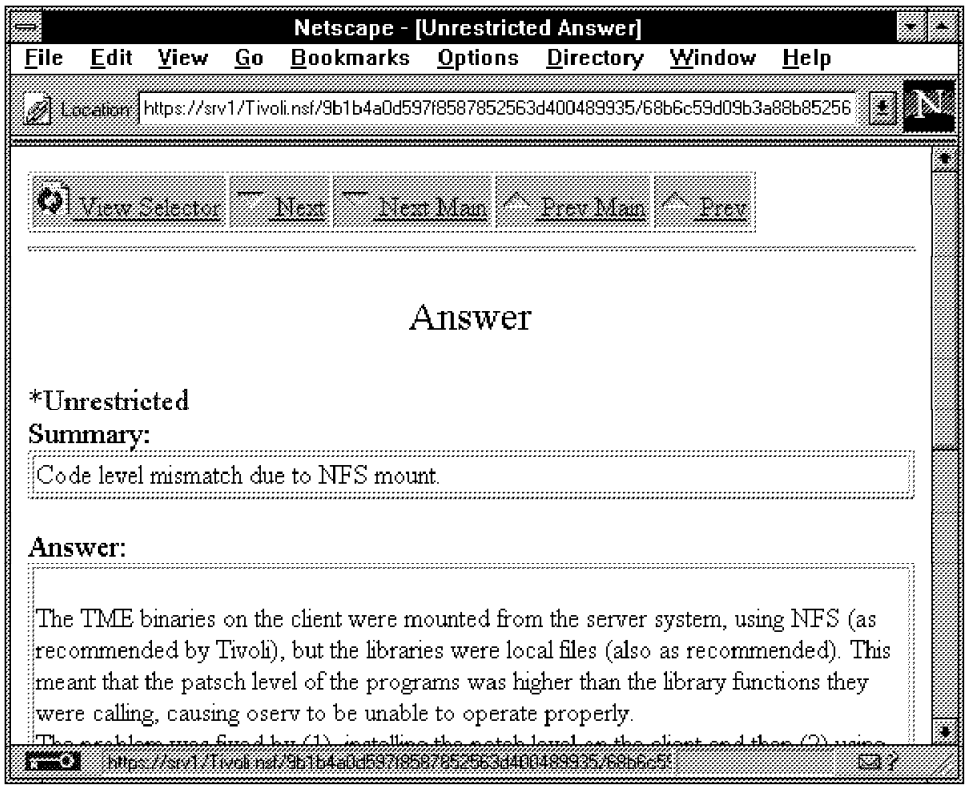


Figure 113. Unrestricted Answer from Web Browser

The answer forms are designed to be entered only as a response to a question. They inherit values from the parent document. To answer a question an administrator must click on the appropriate button while in the question document. Clicking on the Restricted Answer button will cause the session to switch to SSL.

The following actions defined in 6.9, “Actions” on page 119 are active:

- Next
- Next Main
- Prev Main
- Prev
- View Selector

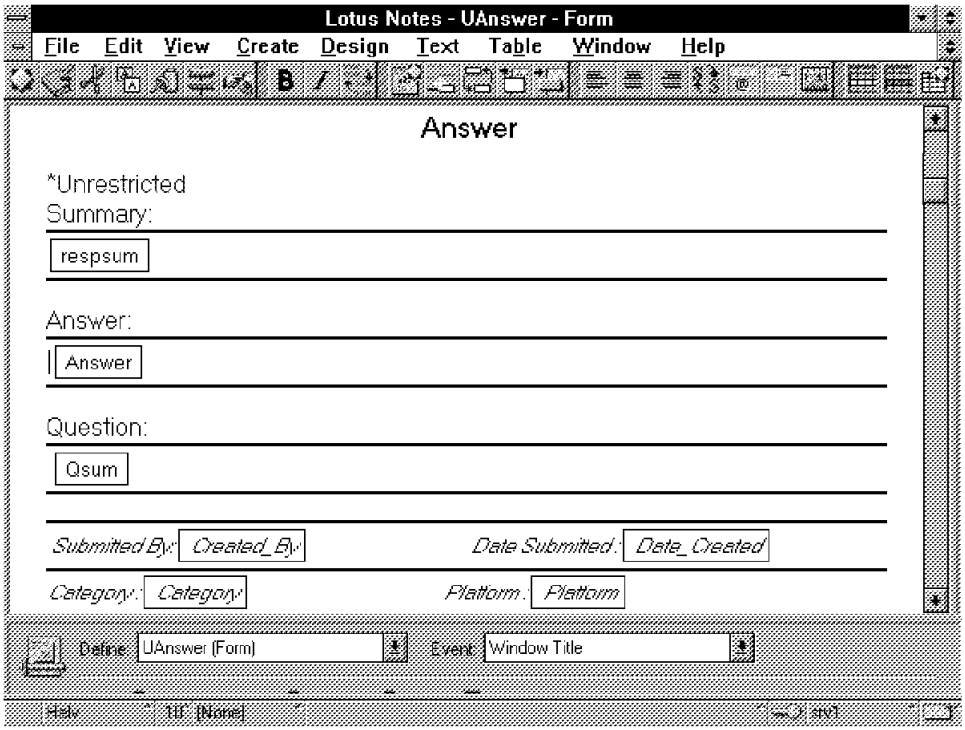


Figure 114. Answer Form Design

Table 11. Restricted and Unrestricted Form Design

Field	**Type**	**Default Value or Value**
respsum	Text - Editable	
Answer	Text - Editable	
Qsum		Qsum (inherited from parent doc)
Created_By	Computed when composed	@Name([CN];@UserName)
Date_Created	Computed when composed Time (show date)	@Now
Category	Keywords (TME 10, etc.), Editable Allow multi-values Allow values not in list	Category (inherited from parent doc)
Platform	Keywords (AIX, MVS, etc.), Editable Allow multi-values Allow values not in list	Platform (inherited from parent doc)

6.8.3 Discussion Forms

The Discussion form is a document type form. Discussion documents are threaded together by the user's choice of platform and category. Discussion threads are not related in a document/response format. Each discussion document has a category and platform the user must fill in. The Discussion view sorts the documents according to category, platform, and date created. The category and platform fields will inherit values from a selected document. This means a user wishing to continue a discussion will automatically have the correct category and platform fields if the relevant document is highlighted or the new discussion record is entered by clicking the Discuss button in an existing document.

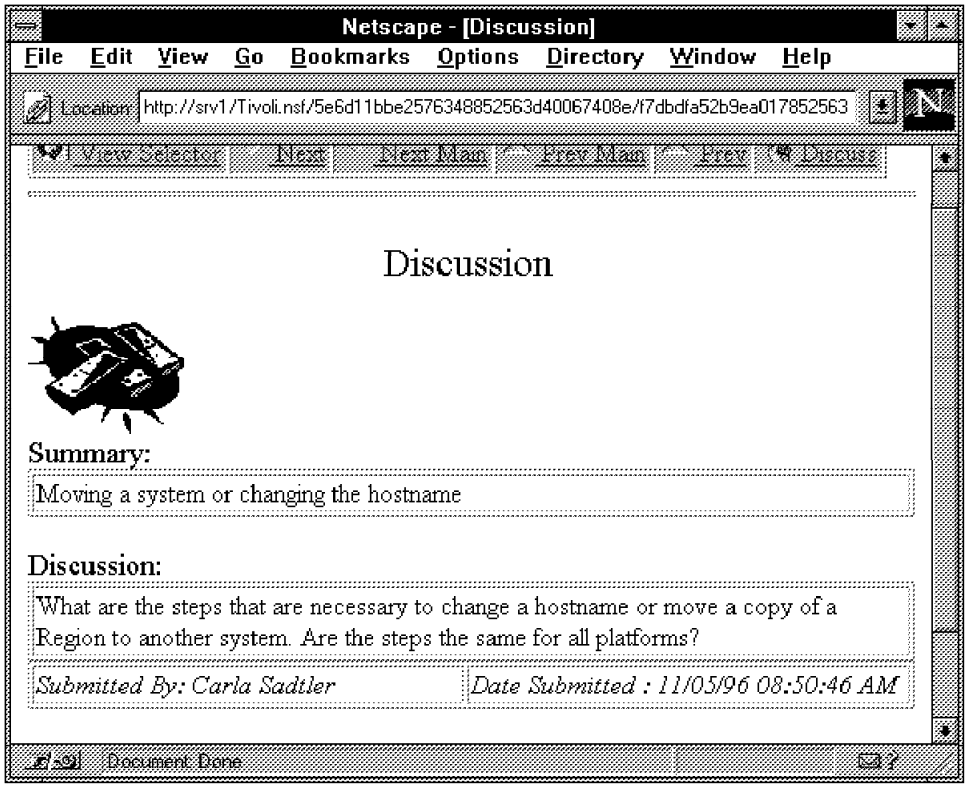


Figure 115. Discussion Document

Discussion forms are protected at the form properties security level to allow administrators, trusted users, submitters, and subscribers (including Anonymous users) to read documents with this form. Only administrators, trusted users, and submitters are allowed to create documents with this form.

The following actions defined 6.9, "Actions" on page 119 are active:

- Next
- Next Main
- Prev Main
- Prev
- View Selector
- Discuss

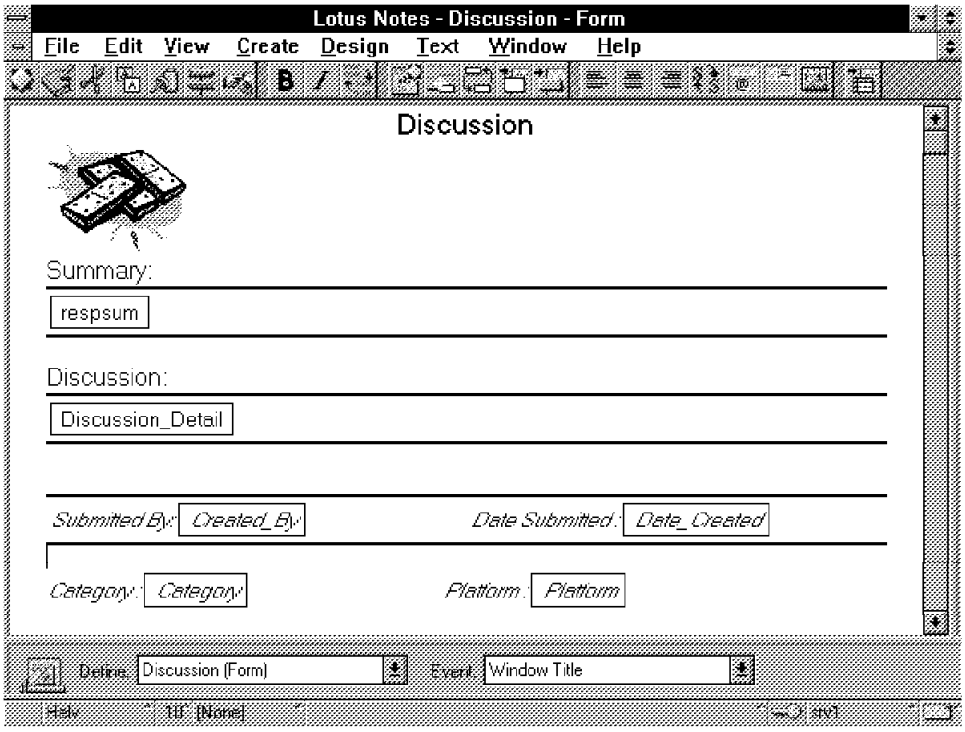


Figure 116. Discussion Form Design

Table 12. Discussion Form Design		
Field	**Type**	**Default Value or Value**
respsum	Text - Editable	
Discussion_Detail	Text - Editable	
Created_By	Computed when composed	@Name([CN];@UserName)
Date_Created	Computed when composed Time (show date)	@Now
Category	Keywords (TME 10, etc.) - Editable Allow multi-values Allow values not in list	Category (inherited from selected doc)
Platform	Keywords (AIX, MVS, etc.) - Editable Allow multi-values Allow values not in list	Platform (inherited from selected doc)

6.9 Actions

Several actions were used throughout the design in both the forms and views. Some are hidden based on the user's authority by using the @UserRoles command to check the user against a list of roles authorized to use the command. Note that as a Notes user this works all the time because your user ID is always known. However if you are a Web user and have not gone through a sign in process, you will see the buttons available only to Anonymous users.

Table 13. Action Buttons

Action	Formula	Hide When **
View Selector	@Command([OpenNavigator];"View Selector")	
Ask Question	@Command([Compose];"Question")	
Discuss	@Command([Compose];"Discussion")	
Unrestricted Answer	@Command([Compose];"UAnswer")	@IsNewDoc | @IsNotMember("[Administrators]";@UserRoles)
Restricted Answer	@URLOpen ("https://srv1/Tivoli.nsf/RAnswer?OpenForm")	@IsNewDoc | @IsNotMember("[Administrators]";@UserRoles)
Next	@Command([NavigateNext])	
Next Main	@Command([NavigateNextMain])	
Prev	@Command([NavigatePrev])	
Next Main	@Command([NavigatePrevMain])	

The hide formulas decide whether a user should see a button based on that user's roles. Since we don't force the user to log on first, this means that some buttons may not be seen by a user because they aren't using their true ID. They are not prompted to log on until they try a restricted command.

Chapter 7. Lotus Notes Firewall Configurations

A Notes server is normally inside a private enterprise network. Questions of the security of the server itself are therefore not a high consideration, often limited to making sure that it is physically secure and that passwords are non-trivial, for example. In the Internet, where any system may be the target for crackers, protection of the server is all-important.

In fact, although the objective is to protect the server, it is really the systems *behind* the server that need to be protected. An Internet server is, by its nature, exposed to some degree of abuse. However, by opening a connection to the Internet you do not want to also open up your private network to attack. Providing this kind of protection is the job of a firewall.

7.1 Firewall Configurations

There is no typical firewall configuration, because in practice an Internet firewall may perform a number of functions, for example:

- Limit access from the Internet so that only the services that you intend to offer can be accessed.
- Provide controlled access to Internet services for users within a private network.
- Perform a gateway role for applications that store and forward information, such as e-mail and news services.
- Provide an encrypted tunneling capability to pass application traffic securely across the Internet.
- Act as a screen to hide or disguise the real content of a private network.
- Act as a gateway between different IP network addressing schemes. (This is not really a security function, but it is something that a firewall is frequently called on to do, because of limitations on the number of available legal IP address ranges.)

This is a formidable list of roles and the result is that no two firewall implementations are alike. However, there are a number of element that are common to most firewall designs.

7.1.1.1 Packet Filtering

Packet filters intercept IP packets in transit and test them against a set of filter rules. This forms the most basic protection mechanism. In fact, filtering is often performed by a screening router, rather than a dedicated firewall machine. Packet filters either allow a packet to pass on to its destination or block it, based on criteria such as:

- The source and destination IP addresses
- The origin of the packet (Did it come from the Internet or from a private network?)
- The client and server port numbers
- The session-level protocol carried by the packet (UDP, TCP, ICMP etc.)

© Copyright IBM Corp. 1997

It is a good idea to employ a packet filter as a screen in front of a World Wide Web server even in cases where the server is not connected to a private network. Crackers will employ tools such as port scanners to probe a server for unsecured services or known application bugs. By placing a screening filter in front of the server you can frustrate this kind of attack. For example you could place a Web server behind a screening router which prevents access to all except the Web server port (TCP port 80).

Unfortunately, packet filtering only provides a full solution in very limited cases. Crackers can use spoofing techniques to bypass some of the protection that a filter offers. Although this may not allow access to restricted services, it can give them a window into the private network that a firewall is supposed to hide. In most cases, therefore, filters have to be combined with some kind of proxy service for a fully secure environment.

7.1.1.2 Proxy Services

A proxy service is a firewall function that accepts session requests from one side of the firewall, checks whether they are permissible and, if so, passes them on to the target system. In addition to allowing effective authorization checking, a proxy also breaks the session at the firewall boundary. This means that an attacker cannot trace the session back to its source, thereby greatly reducing the size of his or her target.

Proxy services are either *application level* in which case they have some knowledge of the application protocols, or *session level* in which case they relay the underlying connection and are transparent to the application.

7.1.1.3 Gateway Services

A firewall is often required to act as an application gateway for services that use the Internet. For example, a firewall may act as an SMTP mail gateway and a DNS server. The objective of these services is to provide access to these Internet services for local users, but to only reveal the minimum necessary information about local users and systems to the rest of the world.

7.1.1.4 System Protection, Logging and Auditing

A firewall is likely to come under attack at some point. It needs to be strongly constructed to resist break-in attempts. It should also provide good logging and auditing facilities, so that attack attempts can be detected and repulsed.

7.1.2 Putting it Together, the DMZ Configuration

We have said that there are many possible firewall designs, depending on the requirements being addressed. However, most mid-to-large sized enterprises have common requirements which lead to firewall configurations that are generally similar to each other.

The common requirements are:

- Provide secure access to Internet services for users within the enterprise. This is at least Web (HTTP) access and FTP, and any number of other services (NNTP, Telnet, etc.).
- Provide servers that can be accessed by external Internet users. Usually at least a Web server.
- Provide a gateway for Internet mail.

The recommended solution to these requirements is a demilitarized zone (DMZ) configuration. Figure 117 on page 123 shows how this is constructed. The DMZ is a network that is neither inside the private network nor part of the Internet. IP filters are employed to screen this network. They are designed to allow inbound sessions to reach the services we are offering, but still provide some protection. Between the DMZ and the private network lies a more solid boundary. This employs proxy servers and application gateways to separate the two networks, making the inside invisible from the outside, but nonetheless allowing local users access to the outside world.

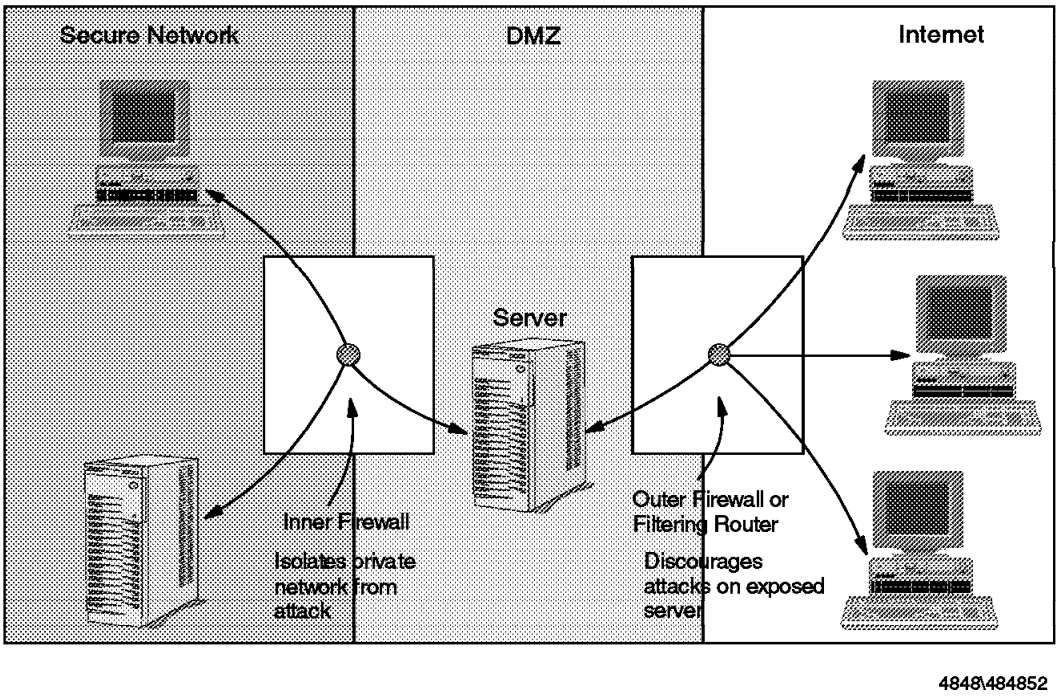


Figure 117. A DMZ Firewall Configuration

7.2 Positioning a Domino Server within a Firewall

If we are using Domino to provide access to Notes databases from the Internet, we must give some thought to how to place the server within a firewall configuration for access across the Internet.

In fact, from the point of view of a client on the Internet, a Domino server is just another World Wide Web server, so we can reasonably apply standard Internet practices for positioning and protecting it, which means that the best configuration is within a DMZ. This still leaves a number of configuration options for implementing Notes within a firewall to explore:

1. How to access and update the Domino server's databases from within the private network.
2. How to use Notes across the Internet (using regular Notes clients rather than Web browsers).
3. How to exploit the non-IP communication capabilities of Notes to eliminate some firewall requirements.

4. How to configure Notes as a Simple Mail Transfer Protocol (SMTP) gateway.

We have said that a Domino server can be viewed as simply another World Wide Web server, supporting HTTP and SSL protocols. This means that the service placement criteria that we described in 7.1, “Firewall Configurations” on page 121 apply. In summary these criteria resolve as follows:

- A server that is accessible from the Internet should never be within a secure network. Instead it should either be directly connected to the Internet, or in a demilitarized zone (DMZ) between the Internet and a private network.
- The server is, by definition, accessible from any Internet client. It is therefore open to attack and should be treated as an extension of the firewall. Such an exposed server could be thought of as being sacrificial. That is to say, you will do your best to protect it and prevent a cracker from gaining access to it. However, there is always a risk that it will be compromised. Your firewall design should minimize the damage that can be done in the event of this happening.

One headache presented by the classic DMZ solution is that of maintaining the security of a server, while at the same time keeping it up-to-date. This is a problem even for servers that simply publish slow-changing information. It is even worse if the server is publishing data that changes in real time, or is receiving and processing input from client users. The problem is often surmounted by opening up a number of *wormholes*: connections through the internal firewall that support maintenance sessions such as FTP, Telnet and database connections. From a cracker′s point of view, each new wormhole represents a potential break-in point.

Domino gives us an excellent solution to this knotty problem. By using Notes′ replication capabilities we can maintain the data published on and received from the Web on a server located safely in our secure network. We still have to have a wormhole to pass data between the two servers, but the connection it provides can be authenticated and encrypted, making it very difficult for a cracker to exploit. In fact it does not even need to be a TCP/IP connection. What′s more, all administrator access to the Notes databases is also controlled using strong authentication methods, which further improves security. We further recommend setting up any server that is in a non-secure network as a separate Notes domain, so that if it does get compromised it reveals the minimum about the rest of the Notes environment.

Figure 118 on page 125 shows how Domino is used in a typical DMZ configuration.

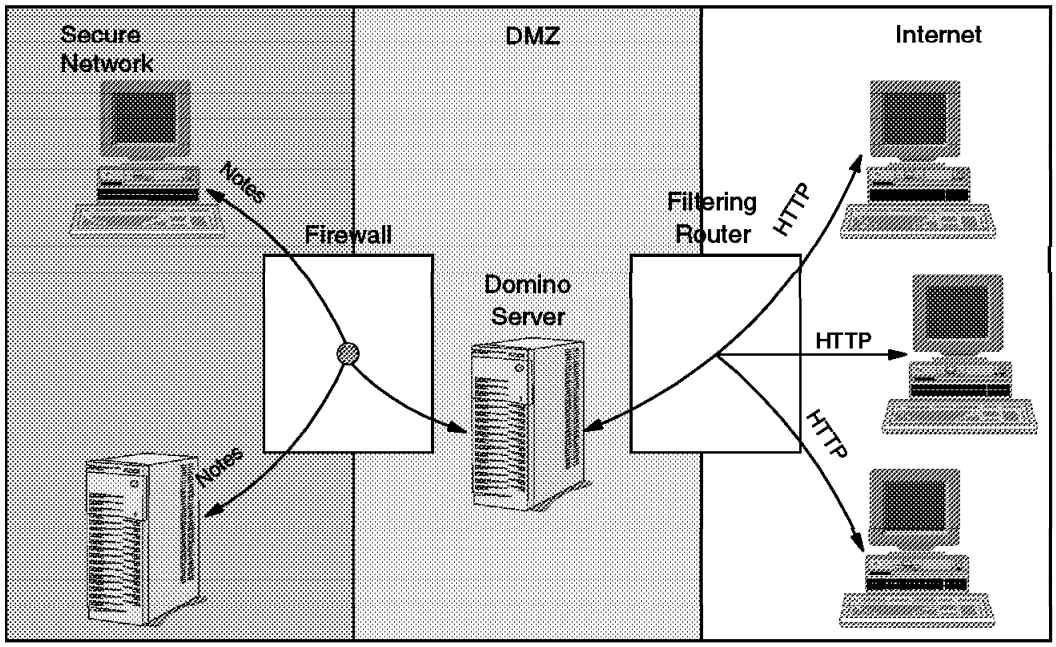


Figure 118. Domino in a DMZ Configuration

A full description of Notes replication is beyond the scope of this book, but we give a brief summary of the aspects that impact firewall configuration below. Note that we have ignored all the subtleties of the process, such as how database records are selected and how replication conflicts are handled. You should refer to *Lotus Notes Network Design*, SR23-7378 for a more detailed treatment of this topic.

7.2.1.1 A Summary of Notes Replication Methods

Replication, as the name suggests, involves making two copies of a Notes database the same. In fact, it is a kind of merge operation in which updated database records are selected from the source database server and replaced on the target database server. There are four replication modes, as illustrated by Figure 119 on page 126. The figure shows a single database that is replicated across two servers. Two documents have been updated in the database, one on Server 1, the other on Server 2 (A′ represents an updated version of document A). In each case the replication is initiated by Server 1, either manually or as a result of a scheduled action.

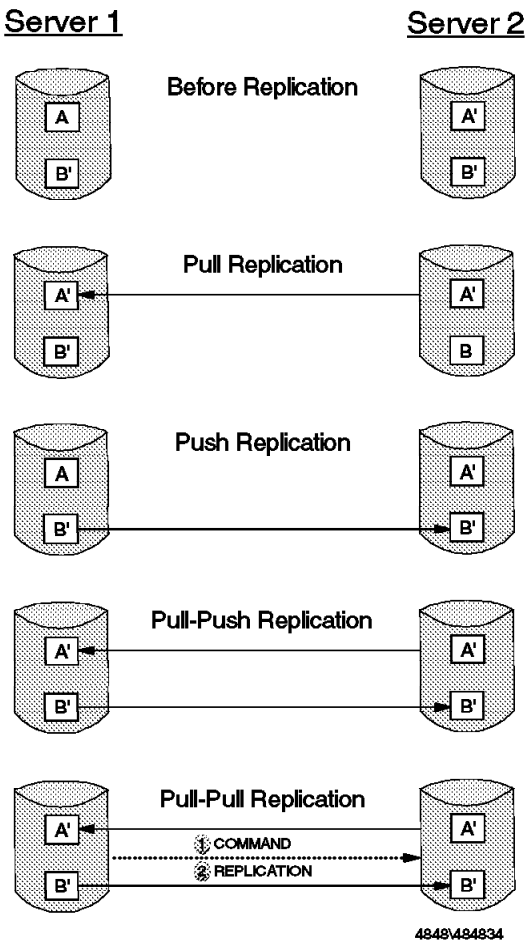


Figure 119. Notes Replication Modes

The first two replication modes, PUSH and PULL, are *one-way* operations. After the replication, only the target system contains the fully-updated database. PUSH and PULL are typically used where one server maintains the master copy of a database and the other just echoes it. The PULL-PUSH replication mode is a combination of the two previous methods. In this case, both servers will be synchronized following the replication. The final replication mode, PULL-PULL, is rather more complex. The end result is that both servers are synchronized, just as for PULL-PUSH. However, the actual update process is done using two PULL requests, one from the initiating server, the other from the second server, but executed as a result of a request from the initiating server.

7.3 Examples of Configuring a Firewall for Domino

In the recommended DMZ configuration, there are two firewall elements that need to be configured: the screening filter that lies between the server and the Internet and the inner firewall that lies behind the server, between it and the secure network.

In this section we describe the requirements for the outer filters and then show examples of a number of different ways to configure server replication across the inner firewall.

7.3.1 DMZ Configuration: Screening Filter Requirements

The objective of placing an IP filter between a server and the Internet is to limit the scope for an attacker to probe the server system. Typically the session will pass through a local IP router anyway, so normally the filtering capability of the router is employed. The desirable filter characteristics to protect a Web server such as Domino are listed in Table 14.

Table 14. Requirements for Screening Filters

Filter Requirement	Details
1. Allow sessions only to the publicly accessible servers	The filters should reject session requests for any IP addresses other than the addresses of the servers that you want to be accessible from the Internet.
2. Allow sessions only to the publicly accessible IP services	In addition to restricting session requests to the publicly accessible servers, the filters should also reject requests for anything other than the IP services that you want to offer. For a Domino server these are: • TCP port 80 for HTTP (normal Web access) • TCP port 443 for SSL (secure Web access) • TCP port 1352 for Lotus Notes access (if you want Notes clients or servers from the Internet to have access to your server)
3. Prevent IP address spoofing	The basic filters described above rely on extracting the source and destination addresses from each IP packet. But an attacker can send packets that appear to come from one of your trusted IP networks and try to gain access. The filters should reject such packets if they appear on the Internet side of the screening router.
4. Prevent well-known port misuse	Crackers sometimes use a technique called *source porting* to try to evade IP filters. This means using a well known port which is normally a server port as a client port (for example, using TCP/80 as a source port). Directional TCP filters can prevent this. We describe these filters in 7.3.1.1, “Directional TCP Filtering” on page 128.
5. Log packets that are rejected by the filter rules	If someone is attacking your site you want to be able to detect it as early as possible. Thorough logging is one of the weapons that helps this detection.

Note that different routers have more or less sophisticated filtering functions, so it may not be possible to implement everything in the table. What level is acceptable to you depends on what the cost of a server break-in would be.

What Does a Break-in Cost?

If a cracker makes a mess of your Web server the cost can be very difficult to assess. In some cases there is a direct financial loss, but more frequently the cost is in terms of perception and reputation. For example, recent well-publicized attacks on U.S. defense department Web sites did not (reportedly) affect national security in any way. However, they did nothing to enhance the reputation of the defense establishment. The cost of this perception could run into millions of dollars in the next congressional budget debate.

7.3.1.1 Directional TCP Filtering

To prevent an attacker from using a well-known port as a source port, filters exploit a feature of the TCP/IP session setup handshake. When a client tries to establish a new TCP session it sends an SYN packet and the server responds with its own SYN packet, which has the acknowledgement (ACK) flag set. From that point in the session, each packet contains an ACK. To put it another way, the only packet in the session that does not have an ACK is the initial session request.

IP filters can therefore look for the ACK flag and use it as an indicator of the direction in which a session was initiated. Figure 120 illustrates how such a filter can frustrate an attacker using a well-known port (in this case the Notes server port) as a source port.

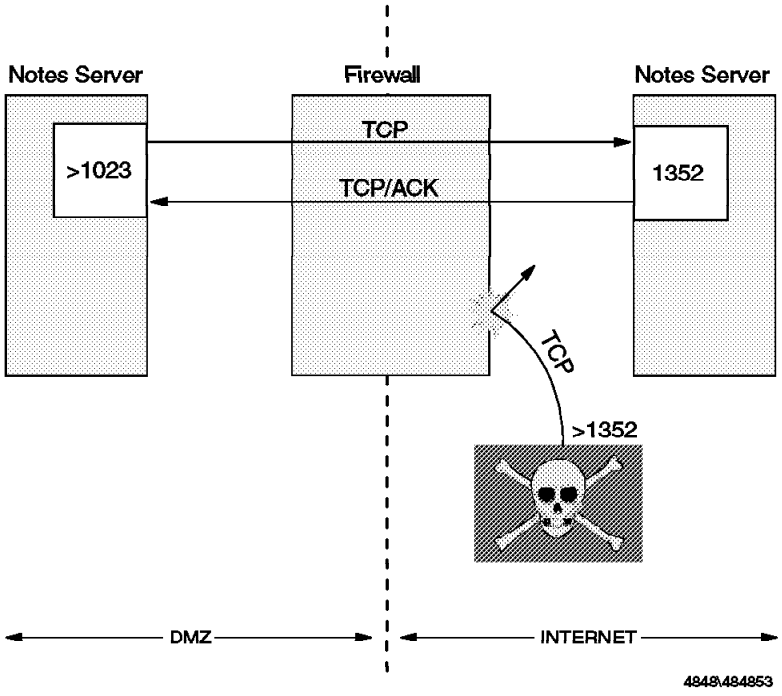


Figure 120. Directional Filtering Using TCP/ACK Testing

7.3.1.2 How Much Protection Do Filters Give?

In this section we have been discussing the first line of defense, screening IP filters. These make the cracker′s life more difficult, but they are only a part of your armor. Filters do nothing to prevent some common modes of attack, such as:

- Attacks that exploit some bug or administrative error in the server itself. The best way to counter this threat is to be vigilant in setting up the server and to monitor sources of information about common problems, such as the World Wide Web security FAQ (at http://www.genome.wi.mit.edu/WWW/faqs/www-security-faq.html).
- Denial of service attacks, intended to bring your server to its knees. These include bombarding it with unconsummated SYN packets and sending malformed IP packets that the server cannot handle (such as over length ping packets).

7.3.2 DMZ Configuration: Inner Firewall Requirements

As we have said, there are a number of different options for configuring the inner firewall. In each case we want to pass Notes sessions between the Domino server and systems in the secure network (either server-to-server sessions for database replication or client/server sessions for manual configuration of the Domino server). Our first recommendation is to make sure that all of these sessions can only be initiated from inside the secure network. Otherwise, if a cracker breaks into a system within the DMZ, he or she can use it as a base to attack the secure network (see Figure 121 and Figure 122 on page 130).

Multiple Layers of Defense

In fact, even if someone did try to abuse the Notes connection in this way, Notes authentication would make life very difficult. Nonetheless it is not a good idea to invite an attack. In firewall design you have to assume that *somehow* an attacker with enough time and brain-power can get through your defenses. By providing multiple layers of defense, you increase the chance that he or she will give up and apply his or her talents to an easier target.

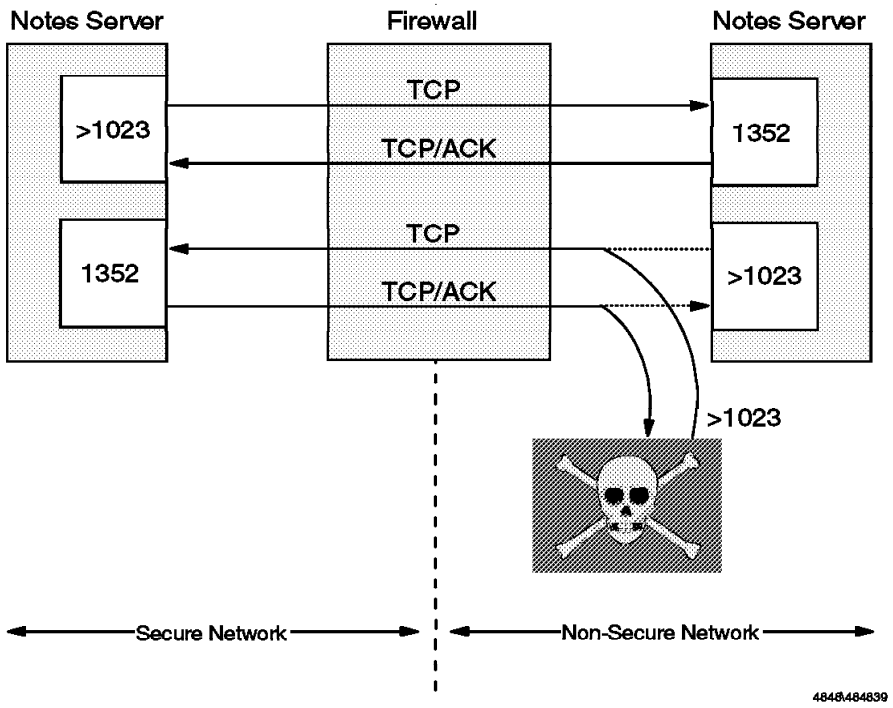


Figure 121. Two-Way Firewall Configuration and Enemies Outside

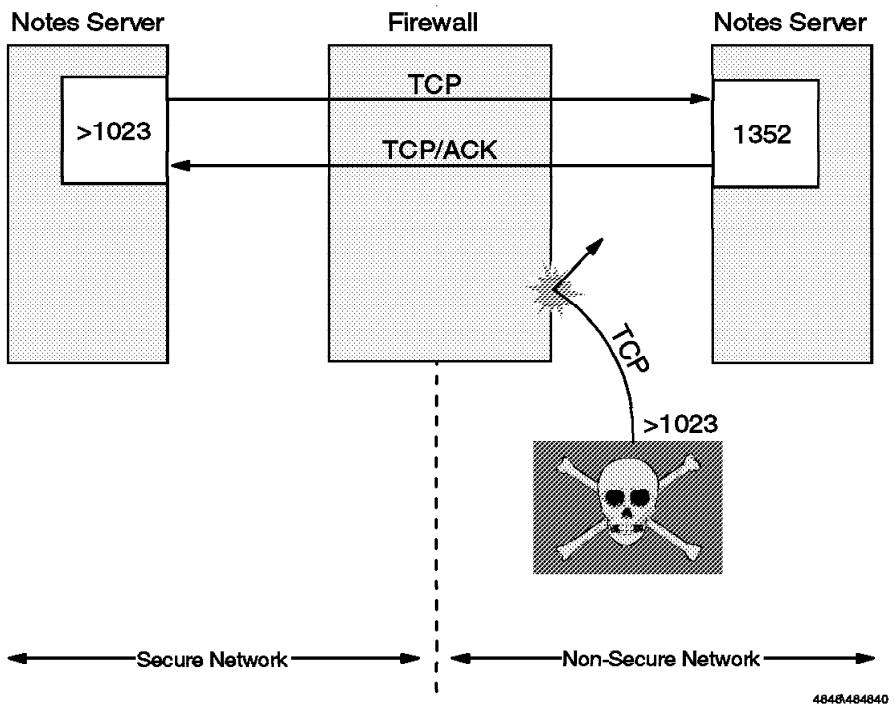


Figure 122. Why a One-Way Firewall Is Better

This requirement limits the ways that we can set up database replication. We need the initial session request for replication to come from the secure network end. PUSH, PULL and PULL-PUSH replication can be used in this way, but PULL-PULL replication will not work, because the pull requests are invoked from each end of the session (see 7.2.1.1, “A Summary of Notes Replication Methods” on page 125). Figure 123 to Figure 125 on page 131 illustrate which Notes sessions are permissible and which are not.

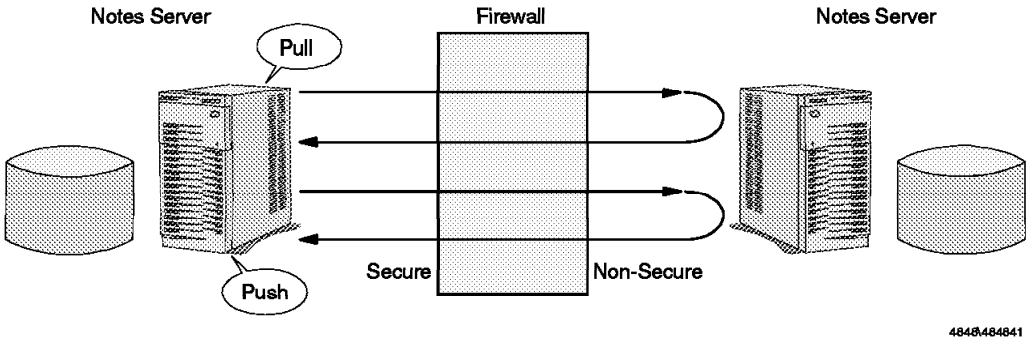


Figure 123. Pull and Push Replication and One-Way Firewall

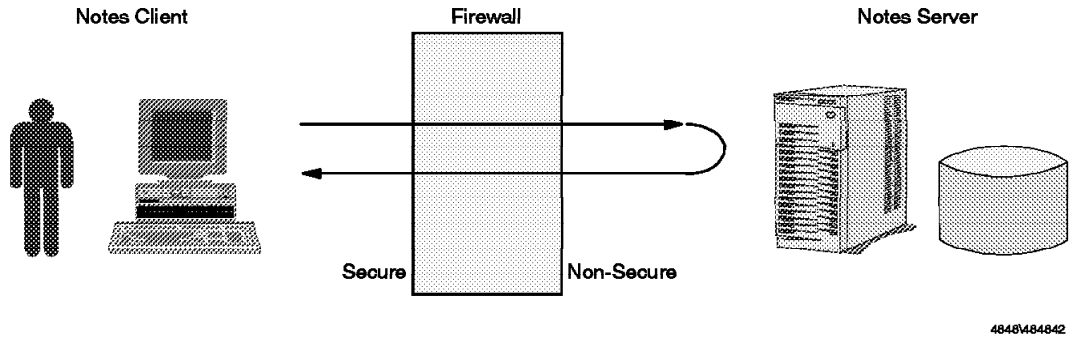


Figure 124. Notes Client Access and One-Way Firewall

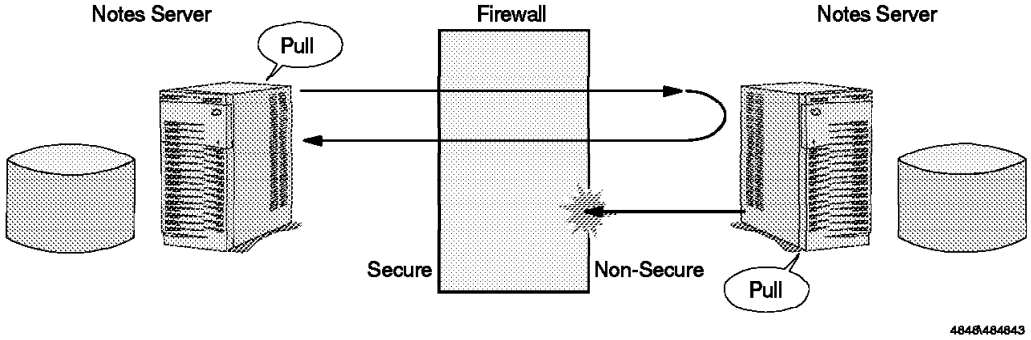


Figure 125. Pull-Pull Replication and One-Way Firewall

7.3.2.1 Configuration Examples

We look at five different configurations for replicating between a server in the secure network and a Domino server in the DMZ:

1. Filtering sessions through the internal firewall
2. Using SOCKS to pass sessions across the internal firewall
3. Using a proxy to pass sessions across the internal firewall
4. Using Network Address Translation across the internal firewall
5. Using a non-IP Notes connection

In each example we use the IBM Internet Connection Secure Network Gateway product for the firewall (at Version 2.2 level). We used a simple lab environment to test each configuration, as shown in Figure 126 on page 132.

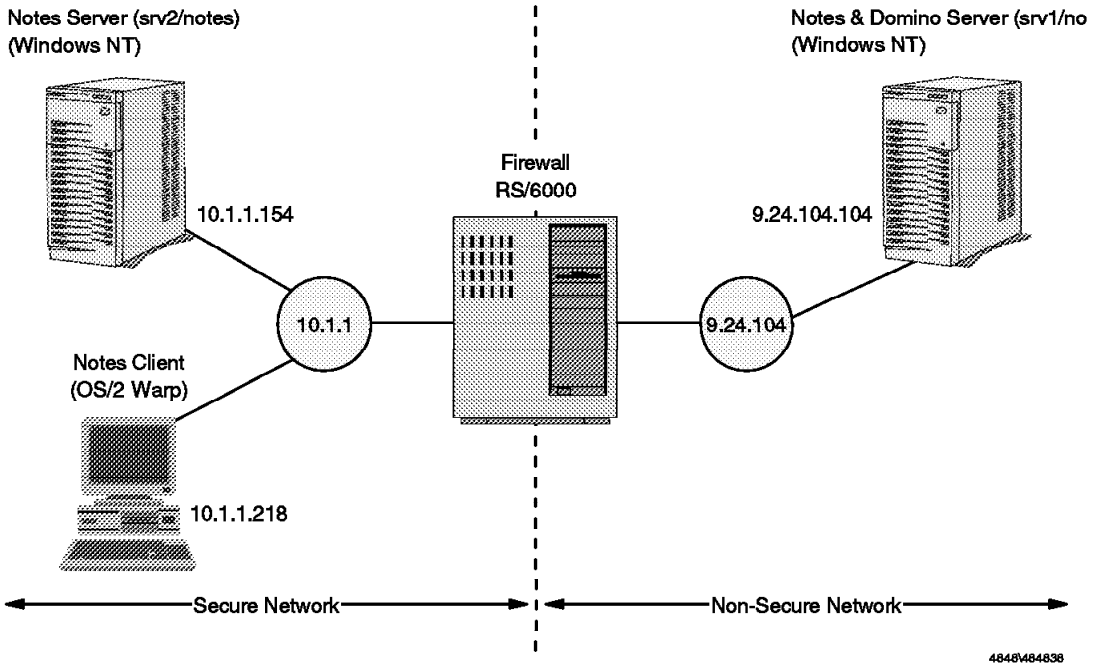


Figure 126. Our Lab Configuration

7.3.2.2 Example 1: Filtering Notes Sessions through a Firewall

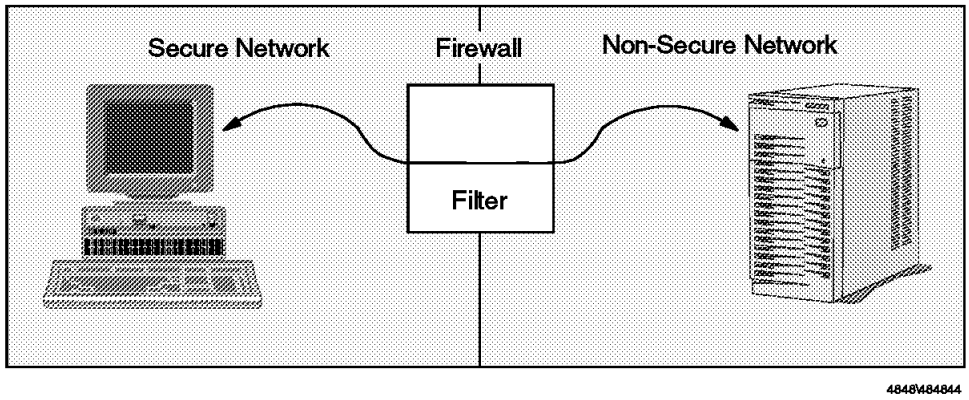


Figure 127. Filter Firewall

This is the simplest approach, just passing the Notes sessions directly through the firewall, using IP filters to limit the possible connections. Using filters in this way is not generally recommended, because it is better to use a relay or proxy application to break sessions at the firewall. However, with the application level authentication that Notes provides, filters are an acceptable method. Filters are also used in combination with the other techniques we describe below.

A more detailed view of the connection is shown in Figure 128 on page 133. The session is initiated from a server or client inside the secure network. It uses a local TCP port number in the unprivileged range (>1023) and connects to the Notes well-known port on the server in the DMZ or Internet (TCP/1352).

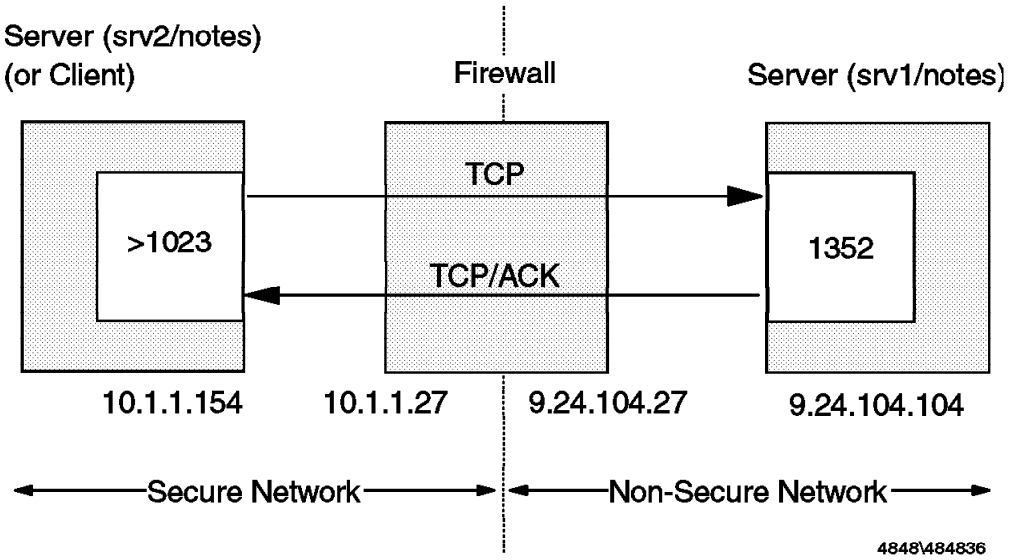


Figure 128. One-Way Filter Firewall

In SNG, the filters are controlled by a filters configuration file, /etc/security/fwfilters.cfg. Normally you would use the AIX Systems Management Interface Tool (SMIT) dialog to create the entries in the file. Each line in the file represents a filter rule. Although SMIT can help you a lot, you should be familiar with these rules. This will help you use them efficiently.

When an IP packet is received by either interface of the firewall, it will check the header information against each rule in the configuration file. It starts at the top and works down until it finds a rule that matches the packet. Then it takes the action indicated by the rule, either to allow the packet to pass or to block and discard it. There is a default rule at the end of the file that blocks every packet that has not yet matched a rule.

If you use SMIT to configure the filter rules you will find that there are two options, to create predefined filter sets or to generate individual rules. The predefined sets create a group of filter rules for a specific IP configuration (for example, HTTP access from the secure network to the non-secure network). There is no option for Notes among the predefined sets, but the approach we tried was to use the SMIT dialog to create a set of rules for a similar application configuration and then edit the resulting fwfilters.cfg file. In SMIT, select **Internet Connection Secured Network Gateway (SNG)**, then **Filters**, followed by **Predefined Filter Rule Sets**, and then select **Add**. You will see a lot of predefined filter rule sets. We selected **Permit telnet FROM secure network DIRECTLY TO non-secure network** to create the basis for our Notes rules. Figure 129 on page 134 shows the SMIT screen and Figure 130 on page 134 shows the filter rules after we had edited the file to change the application details from Telnet to Notes.

```
                          Add a Predefined Filter Set

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  Set Number                                       11 - (SN,NN) Permit
* Source Address                                 [10.1.1.0]
* Source Mask                                    [255.255.255.0]
* Destination Address                            [9.24.104.0]
* Destination Mask                               [255.255.255.0]
* Tunnel ID                                           0
  User Description                               []
```

Figure 129. SMIT Screen to Define Predefined Filter Rule Sets

```
#------------------------------------------------------------------------#
#PF (SN,NN) Permit LotusNotes FROM secure network DIRECTLY TO non-secure
 network ->  <-
#------------------------------------------------------------------------#

permit 10.1.1.0     255.255.255.0   9.24.104.104 255.255.255.255 tcp
gt 1023 eq 1352 both route both l=n f=y

permit 9.24.104.104 255.255.255.255 10.1.1.0     255.255.255.0   tcp/ack
eq 1352 gt 1023 both route both l=n f=y
```

Figure 130. Filter Rules to Allow Notes Sessions to Pass

The format of the filter rule file looks rather daunting at first, but in fact it is not too complicated. A file containing the two rules shown above implements the following logic:

- Allow to pass any TCP/IP packet that is going from any node in the 10.1.1 network to IP address 9.24.104.104, *if* the target port is TCP/1352 *and* the source port is in the unprivileged range (1024 and above) *and* the packet is being routed through the firewall. (That is, neither session endpoint is in the firewall itself.)
- Allow to pass any TCP/IP packet that is going from IP address 9.24.104.104 to any node in the 10.1.1 network, *if* the source port is TCP/1352 *and* the target port is in the unprivileged range (1024 and above) *and* the packet is being routed through the firewall *and* the acknowledgement flag is on in the packet. (That is, it is not the first packet of a session.)
- Block any other IP packet from entering or leaving the firewall. (This is the default behavior, no rule needs to be coded for it.)

In order to let the filters rule take effect, select **Validate Rules** from the SMIT dialog to process the filter rule file and load the modified rules into memory.

For a full description of each field in the rule entries and for a large number of examples of rules for different application types, refer to *Building a Firewall with the IBM Internet Connection Secure Network Gateway*, SG24-2577.

7.3.2.3 Example 2: Using SOCKS to Pass Notes Sessions through a Firewall

SOCKS is a session-level proxy mechanism. It requires a modified client system (called a SOCKSified client) which intercepts application calls for TCP/IP services and routes them to a SOCKS server running on the firewall. The SOCKS server then checks that the request is allowed before passing it on to the target server. Figure 131 shows the configuration.

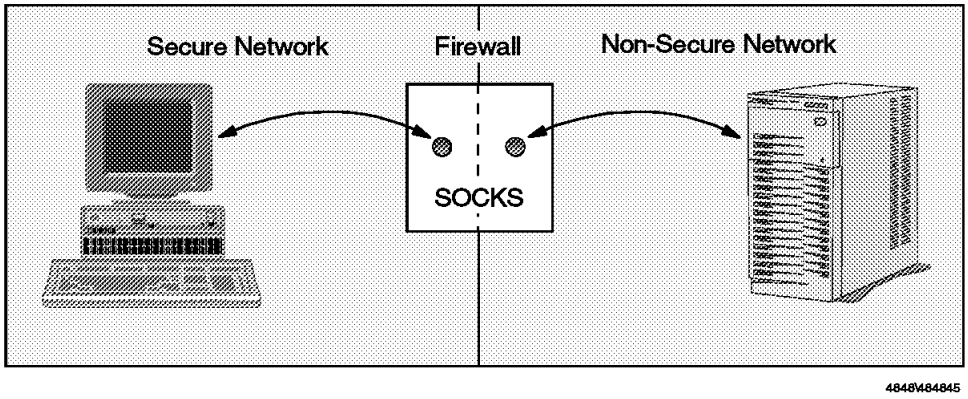


Figure 131. SOCKS Firewall

SOCKS has three advantages over the previous example of routing through the firewall:

1. The server can check in more detail whether a request is allowed or not.
2. From the non-secure network, all requests appear to have come from the firewall directly, so the contents of the secure network are hidden from view.
3. The IP addressing scheme in the secure network does not have to use legal, routable addresses.

For this example we still use IP filters to prevent anything except the SOCKS connection from being established. As you can see from Figure 132, the situation is complicated by having to provide filters for both halves of the session.

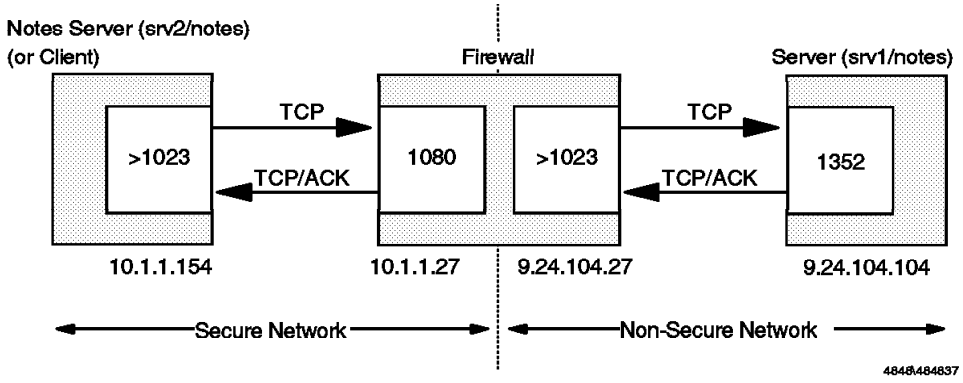


Figure 132. Filter Consideration on Our SOCKS Firewall

In order to use the SOCKS service, we must first configure it. This includes both configuration of the SOCKS server (sock daemon configuration file and filter rule configuration file) and the SOCKS client.

Configuring the SOCKS Server: You can use SMIT to configure the SOCKS server, or you can use an editor to modify the configuration file directly. Enter `smit` at the AIX command line to start SMIT. The SMIT system management menu appears.

In SMIT, select **Internet Connection Secured Network Gateway (SNG)**, then **Socks Services** and then **Add**. You see the screen shown in Figure 133.

```
                    Add Socks Server Configuration Entry

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  Action                                        [permit]
  Employ ident verification                     [None]
  User list                                     []
* Source Address                                [10.1.1.0]
* Source Mask                                   [255.255.255.0]
  Destination Address                           []
  Destination Mask                              []
  Operation Field                               []
  Destination Port Number                       []
  Command to execute                            []
```

Figure 133. SOCKS Server Configuration Screen

Each entry you add in this way builds a rule in the /etc/sockd.conf file that controls which sessions the SOCKS server will handle. It is normally based on IP address, although it can also use the TCP/IP ident service to check for a user ID, if it is supported by the client.

In the first field, specify whether you are permitting or denying access to the hosts. In the remaining fields on this panel, qualify the request and identify the hosts. Then press Enter to generate the updated /etc/sockd.conf file.

The configuration above results in a sockd.conf file that contains the following line:

```
permit   10.1.1.0 255.255.255.0 0 0
```

This file controls access to the sockd server. It contains separate lines to permit or deny access from source to destination hosts. Permission and denial of a service request can be decided based on various combination of the requesting host, the destination host, the destination port number, as well as the requesting user.

When socks server receives a request, it checks against the lines in the file, one line at a time, starting at the top. When it finds a line with conditions that are matched by the request, it stops. The request is either granted or denied based on the action field of that line. If no matching line is found in the entire file, the request is denied.

Configuring Filter Rules for SOCKS: Even though the SOCKS configuration includes rules which look very like filters, these rules are independent of IP packet filter rules. It is important to have IP filters that complement the SOCKS configuration, because the SOCKS server has no concept of secure and non-secure network. It therefore needs IP filters to ensure that the session configuration is what you expect.

The actual filter entries are configured in the same way as the previous example (see 7.3.2.2, "Example 1: Filtering Notes Sessions through a Firewall" on page 132). They are more complex than the previous example because now we are handling two sessions: from the client to the SOCKS server and from the SOCKS server to the real Notes server in the non-secure network. See Figure 134 for the filter rules we used in our example environment.

```
#------------------------------------------------------------------------
# Permit use of Socks FROM secure network TO the firewall ->  <-
#------------------------------------------------------------------------
permit 10.1.1.0 255.255.255.0 10.1.1.27 255.255.255.255 tcp gt 1023 eq 1080
secure local inbound l=n f=y
permit 10.1.1.27 255.255.255.255 10.1.1.0 255.255.255.0 tcp/ack eq 1080 gt
1023 secure local outbound l=n f=y

#------------------------------------------------------------------------
# Permit use of Socks FROM the firewall TO non-secure network -> <-
#------------------------------------------------------------------------
permit 9.24.104.27 255.255.255.255 0 0 tcp gt 1023 eq 1352 non-secure local
outbound l=n f=y
permit 0 0 9.24.104.27 255.255.255.255 tcp/ack eq 1352 gt 1023 non-secure
local inbound l=n f=y
```

Figure 134. /etc/security/fwfilters.cfg File for SOCKS Access

The two sessions in the SOCKS case both terminate at the firewall, so the filters have *Local* specified in place of *Route* (compare the rules with those in Figure 130 on page 134).

Configuring the SOCKS Client: Notes 4.5 supports SOCKS Version 4. We edited the SOCKS proxy field in Notes Server document to reflect the address of the firewall. Figure 135 on page 138 shows this document. The SOCKSified Notes server uses this parameter to determine whether to use a direct or SOCKS connection to a given destination host.

Every time a SOCKSified client has to make a network connection, the session request is passed to the SOCKS server on TCP port 1080 which checks against /etc/sockd.conf before relaying the request.

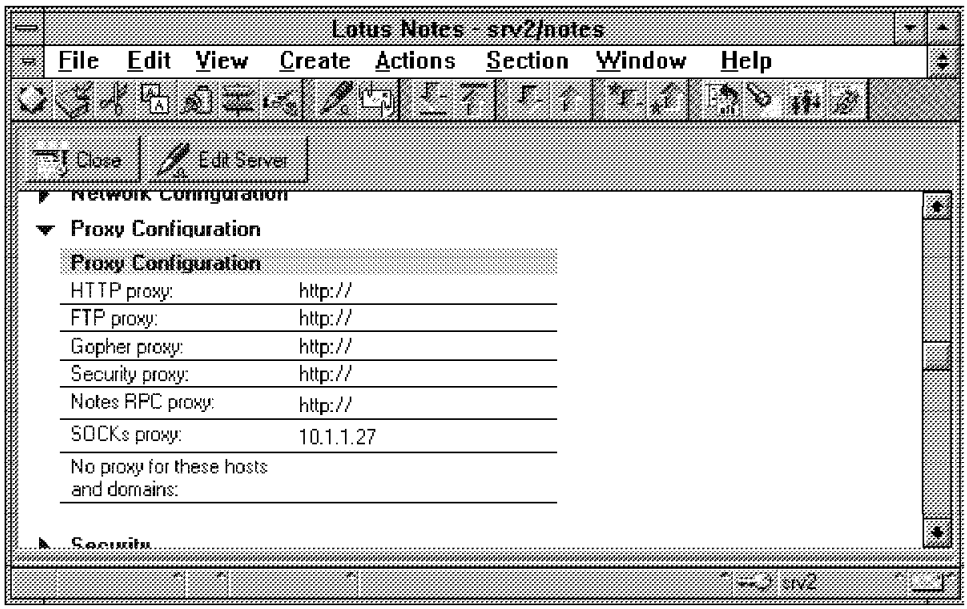


Figure 135. Socks Client Configuration

Note that when using SOCKS, the Notes server that initiates the replication (the secure network server) thinks that it is directly in connection with the real target server. It therefore needs to be able to resolve the server name into an IP address (using /etc/hosts or DNS or a Connection document). The other server thinks that the source server is at the IP address of the firewall. This does not matter, because Notes uses public key-based authentication methods to identify the partner in a replication action and does not rely on network addresses.

7.3.2.4 Example 3: Using a Proxy to Pass Sessions through a Firewall

This example is similar in configuration to the SOCKS case (see Figure 136 on page 139). However, in this case the proxy application on the firewall acts as a relay for just the Notes sessions, rather than the generic function that the SOCKS server provides. This configuration has the same advantages as SOCKS in terms of providing a point of authorization for the session and hiding the secure network from outside scrutiny. It has the additional advantage of not requiring a modified (SOCKSified) client to operate. However, as we will see, it makes the replicating server more difficult to configure compared with the SOCKS case.

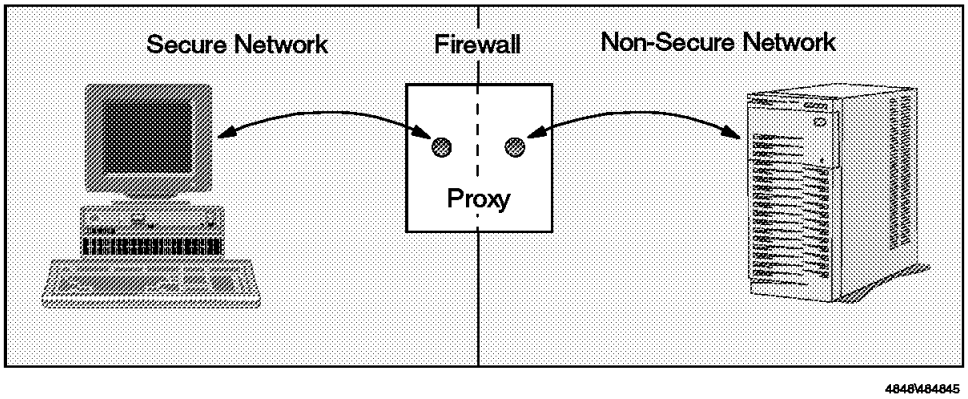


Figure 136. Proxy Firewall

The proxy code itself is a sample called tcp_relay that allows SNG to relay connections between the secure network and the non-secure network. This program was first listed in *Building a Firewall with the IBM Internet Connection Secured Network Gateway*, SG24-2577. This is not a standard part of SNG, so we strongly recommend that you use the SOCKS option instead of this example. However, we include the example here in case for some reason you cannot use SOCKS. (If the Notes server in the secure network is at an earlier level, or if you are using a firewall product without SOCKS support, for example.)

The other possibility that this proxy example gives you is to allow sessions initiated from the non-secure network into the secure network. We have recommended that you avoid this, wherever possible, by using replication initiated from the secure server. However, there may be situations in which you need to allow particular sessions through (for Notes passthru for example).

From an IP filtering viewpoint the proxy configuration is similar to SOCKS, except that the TCP port used by the proxy server is the same as the real server (1352 instead of 1080) as shown in Figure 137.

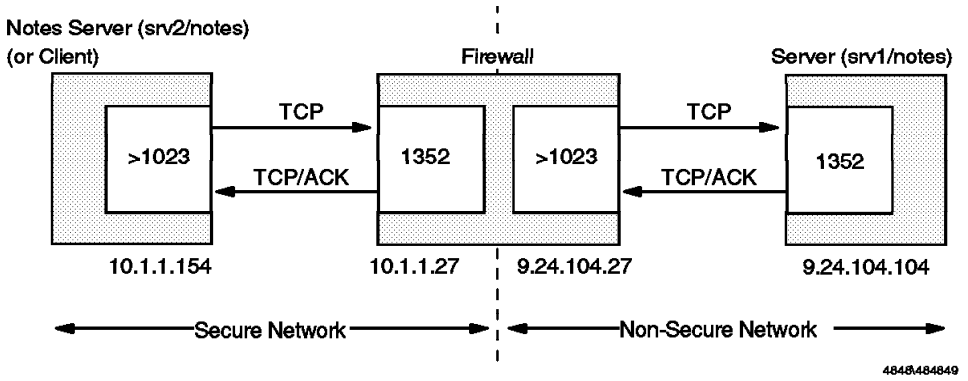


Figure 137. Filter Consideration on Our Proxy Firewall

As for SOCKS, there are three parts to setting up the proxy connection: configuring the proxy server, configuring IP filters to support the proxy server, and configuring the client.

Configuring the tcp_relay Proxy Server: The tcp_relay program uses a simple configuration file to identify which IP address and port combinations it should act as a relay between. Figure 138 on page 140 shows the tcp_relay.cfg file that we used in our sample environment.

```
10.1.1.154      1352    9.24.104.104    1352
9.24.104.104    1352    10.1.1.154      1352
```

Figure 138. tcp_relay.cfg Configuration File

The first entry tells the tcp_relay program to listen on port 1352 for session requests from address 10.1.1.154 and to relay those sessions on to port 1352 on IP address 9.24.104.104. The second line in the file allows us to do the opposite, that is, to start the session from the non-secure network server. Normally you would not need to do this.

To start the proxy application running we have to add it to the inetd daemon startup by adding the following line in /etc/inetd.conf:

```
lotusnote stream tcp nowait nobody /etc/tcp_relay tcp_relay
```

Port IDs and Names

In this case the entry in inetd.conf refers to lotusnote because that is the entry that is placed in /etc/services by the AIX operating system. In other documentation you may see this identified as lotusnotes instead. In fact it does not matter, so long as it is consistent within the one machine, because the port name is internal. Only the port number is used over the network.

Configuring Filter Rules for Proxy: The IP filtering requirements for the proxy setup are very similar to SOCKS. Figure 139 shows the rules that we used.

```
#-------------------------------------------------------------------
# Permit proxy FROM secure network TO firewall -> <-
#-------------------------------------------------------------------
permit 10.1.1.154 255.255.255.255 10.1.1.27  255.255.255.255 tcp
gt 1023 eq 1352 secure local inbound  l=n f=y
permit 10.1.1.27  255.255.255.255 10.1.1.154 255.255.255.255 tcp/ack
eq 1352 gt 1023 secure local outbound l=n f=y

#-------------------------------------------------------------------
# Permit proxy FROM firewall TO non-secure network ->  <-
#-------------------------------------------------------------------
permit 9.24.104.27  255.255.255.255 9.24.104.104 255.255.255.255 tcp
gt 1023 eq 1352 non-secure local outbound l=n f=y
permit 9.24.104.104 255.255.255.255 9.24.104.27  255.255.255.255 tcp/ack
eq 1352 gt 1023 non-secure local inbound  l=n f=y
```

Figure 139. Filter Rules for Proxy Firewall Configuration

Note that in this case the filter rules are more restrictive. (They specify an exact address instead of an address range for the secure network end of the session.) This is because the proxy application can connect sessions *only* between the pair of servers identified in the tcp_relay.cfg file. Note also, that if we wanted to use the relay to allow sessions initiated from the other direction we would need a complementary set of filter rules.

Configuring the Secure Server to use the Proxy: If we want the two servers to replicate through the proxy we have to employ some sleight of hand. The server that initiates the replication is not aware that the proxy exists, so it thinks that the target server is actually at the IP address of the firewall (see Figure 140 on page 141).

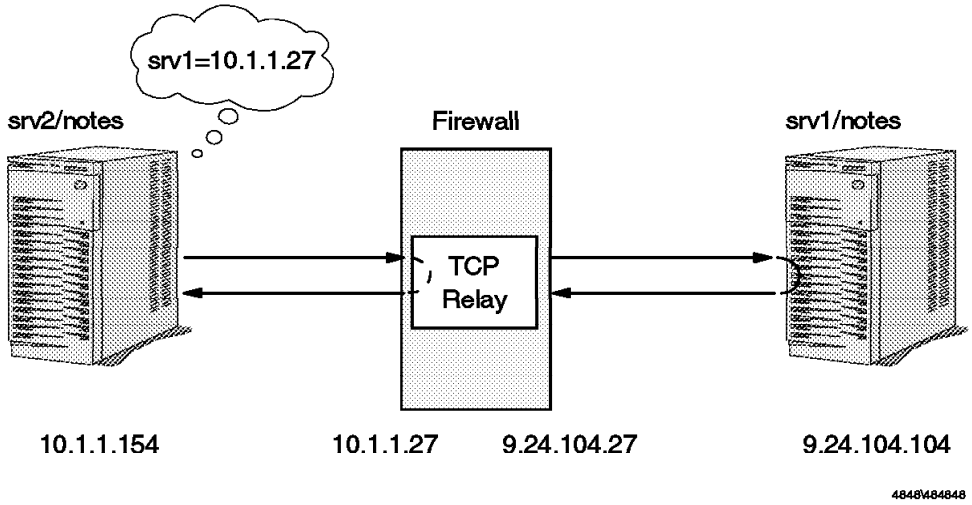


Figure 140. What the Server Thinks

This means that in the Notes Server document network configuration section, Net Address field should be equal to the secure network interface of the firewall. We therefore filled in 10.1.1.27 in the Net Address field. However, the *real* srv1 also has an entry for itself in the Notes Server document. In other words:

- On srv1/notes: srv1′s address is the real one.
- On srv2/notes: srv1′s address is the fake one.

If the two servers belong to the same domain, this will cause some conflicts. All Notes servers in a domain share the same Public Address Book, therefore Notes replication will synchronize the two srv1s definition; the Notes replication mechanism will not allow inconsistency between the two. Of course, this problem also affects Notes clients that belong to the same domain.

To avoid this conflict, you need to set up the Domino server as a separate Notes domain if you want to use a proxy configuration. Then each domain keeps a different Public Address Book and your systems can live with the apparent inconsistency. (In fact we recommend you set up any server that is in a non-secure area or DMZ as a separate domain anyway, to improve security.)

In the secure network, we also had to modify the IP host file, so that srv1 maps to the IP address of the firewall. This requirement may have been due to some feature of our lab environment. Normally you would not expect to have to use a host file, because every server in a domain shares the Public Address Book which contains all the servers′ network addresses.

In fact, all Notes servers act as Notes name servers. Notes clients send name-service query requests to their home server and then the home server translates the Notes server name into its network name or address and sends it back. However, in our Notes 4.5(TB3) on Windows NT environment, we found that the system would not work without the host file.

7.3.2.5 Example 4: Using Network Address Translation to Pass Sessions through a Firewall

Network Address Translation (NAT) is similar to a SOCKS or proxy configuration, in that the session is broken at the firewall boundary. In the SOCKS case, the client needs to be modified so that it can communicate with the SOCKS server and tell it which server to connect to. NAT is more like the proxy server, in that the client does not need to be modified. It sends packets to the real server address and it thinks that the firewall is just another router in the path. However, the firewall alters the contents of the packets that arrive, changing the source address to an address that is outside the firewall boundary. The addresses it uses are drawn from a pool of legal IP addresses. Figure 141 shows this configuration.

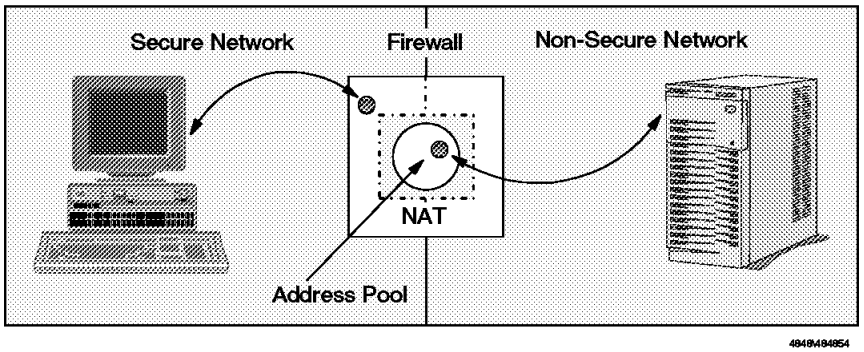


Figure 141. NAT Configuration

Apart from the security advantage gained by breaking the session at the firewall boundary, NAT also allows non-legal IP addresses to be used in the secure network.

To configure NAT for our replication scenario we have to allocate the pool of addresses for NAT to use and then set up a mapping from the secure server address to this address pool. There are four types of entry in the NAT configuration file:

Reserve This allocates a range of addresses to which secure network addresses will be mapped. From the point of view of the server in the non-secure network these addresses are behind the firewall gateway, but really they only exist in the brain of the firewall software.

Translate This defines addresses in the secure network that are to be translated to an address drawn from the reserved pool.

Exclude This allows you to override the translate definition by specifying addresses that you do not want to be translated.

Map The translate entry (above) causes a reserved pool address to be dynamically allocated when a secure network address requests a session through the firewall. The map entry fixes a particular reserved pool address to be associated with a secure network address. Apart from guaranteeing that an address will be available, this also allows you to start sessions from the non-secure side to a specific secure address.

We defined a pool of 64 addresses with one reserve entry and one map entry to allocate a pool address to the secure network server. Figure 142 on page 143 and Figure 143 on page 143 show the SMIT definitions for these two entries.

```
                                Change NAT Entry

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      Entry Fields
  Sequence Number                                     1
  Entry Type                                          RESERVE
* Registered IP Address                               9.24.105.64
* Registered IP Address Mask                          255.255.255.192
  Timeout (in minutes)                                15
```

Figure 142. Reserving the NAT Address Pool

```
                                Change NAT Entry

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      Entry Fields
  Sequence Number                                     1
  Entry Type                                          MAP
* Secured IP Address                                  10.1.1.154
* Registered IP Address                               9.24.105.99 Y
```

Figure 143. Allocating a Fixed Pool Address

To complete the NAT configuration we must create IP filter definitions to allow the session to be established. Figure 144 shows the filter definitions we used. Note that these definitions do not reference the pool of reserved addresses at all. The pool addresses are internal to the firewall and the filters operate outside of them. In other words, the filters are defined as if the session was being passed straight through the firewall.

```
permit 10.1.1.154   255.255.255.255   9.24.104.104 255.255.255.255
tcp     gt 1023 eq 1352 both route both l=n f=y t=0

permit 9.24.104.104 255.255.255.255   10.1.1.154   255.255.255.255
tcp/ack eq 1352 gt 1023 both route both l=n f=y t=0
```

Figure 144. Filter Definitions for NAT Firewall

7.3.2.6 Example 5: Using a Non-IP Notes Connection to Eliminate a Firewall

Firewalls are designed primarily to prevent TCP/IP network packets from the non-secure network reaching the secure network. An extremely good technique for protecting the secure network from unauthorized TCP/IP packets is not to use TCP/IP on your secure network at all. If you have such a network, you can create a configuration in which the Notes server effectively replaces the firewall

altogether. In fact, even if you do have TCP/IP in your secure network, it is possible to provide a number of firewall-like functions without having a firewall installed. For example:

- You can provide a Web service by using a Domino server.
- You can have an e-mail gateway between Notes-based enterprise mail and SMTP.
- You can provide Web browsing, using Passthru and the Notes Web Navigator.

To avoid the need for an Internet firewall, the connection between the non-secure and secure networks should not use TCP/IP. Notes will force all traffic from the non-secure network to be translated into a different protocol. Notes also does all protocol conversions to allow Notes users to access the non-secure network.

Figure 145 illustrates this.

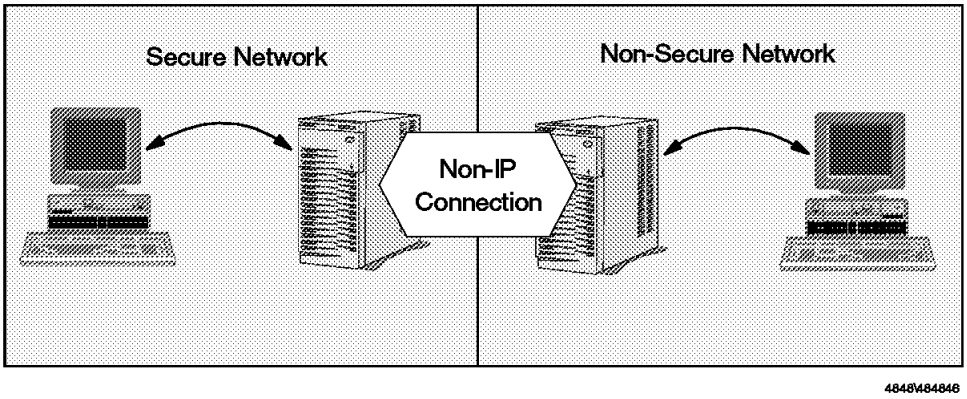


Figure 145. Notes Non-IP Connection Firewall

In this configuration, each server has two network interfaces. One can be a serial connection or a LAN connection running IPX or NetBIOS, the other runs TCP/IP. All communications between the two servers will use the first of these two interfaces.

If a cracker breaks into the non-secure server, the first problem he or she will face is an unfamiliar (non-TCP/IP) network environment. If the secure interface is a LAN connection, it offers some possibility of further attack, because it is a broadcast medium that may allow the cracker to explore other connected systems. However, if it is a serial connection there is no scope for abuse, because it will only handle authenticated Notes traffic between the two servers. A serial link is therefore the recommended approach to take (although it does limit the possible throughput of the connection).

Notes firewalls of this type enable you to filter the traffic going from the secure network to the non-secure network. Because all non-secure network data is translated to and from documents in Notes databases, you can use all of Notes administrative tools to restrict or monitor the information being sent or received.

7.4 Native Notes Access Across the Internet

Up to this point we have been assuming a configuration where access to Notes servers from the Internet is via a World Wide Web client, using Domino. However, TCP/IP Notes connections can also be routed in the same way. We still must recommend that any Notes server visible from the Internet is outside of a secure network, and it should be protected by a screening filter in the same way as a Web server (see Table 14 on page 127).

The superior authentication and encryption facilities of Notes make this an attractive, low-cost, way for two enterprises to share information and e-mail safely.

7.4.1 Using HTTP Proxy Tunneling

When providing Web access for users in a secure network, it is common to install a proxy server. The virtue of this is that the proxy is the only source for Web sessions leaving the secure network, which means that a smaller target is presented to an attacker. Web proxies also allow frequently accessed pages to be cached leading to improved performance.

If you have a Web proxy server installed, you may want to use the same facility to route other traffic, such as Notes connections, into the Internet. Notes 4.5 provides the ability to do this, using a technique called *proxy tunnelling*. It relies on the proxy server supporting the HTTP CONNECT method. This is a protocol designed by Netscape to allow non-Web applications to request a relay connection via the proxy server. The CONNECT method was originally developed to allow SSL connections to be proxied without losing end-to-end encryption, so you will often see it referred to as SSL tunneling.

To set up a Notes server to use a proxy tunnel for replication and mail delivery, do the following:

1. On the server, specify an HTTP proxy URL such as http://9.24.104.27 in the Notes RPC Proxy field, under the proxy section of the server document. (see Figure 146 on page 146).

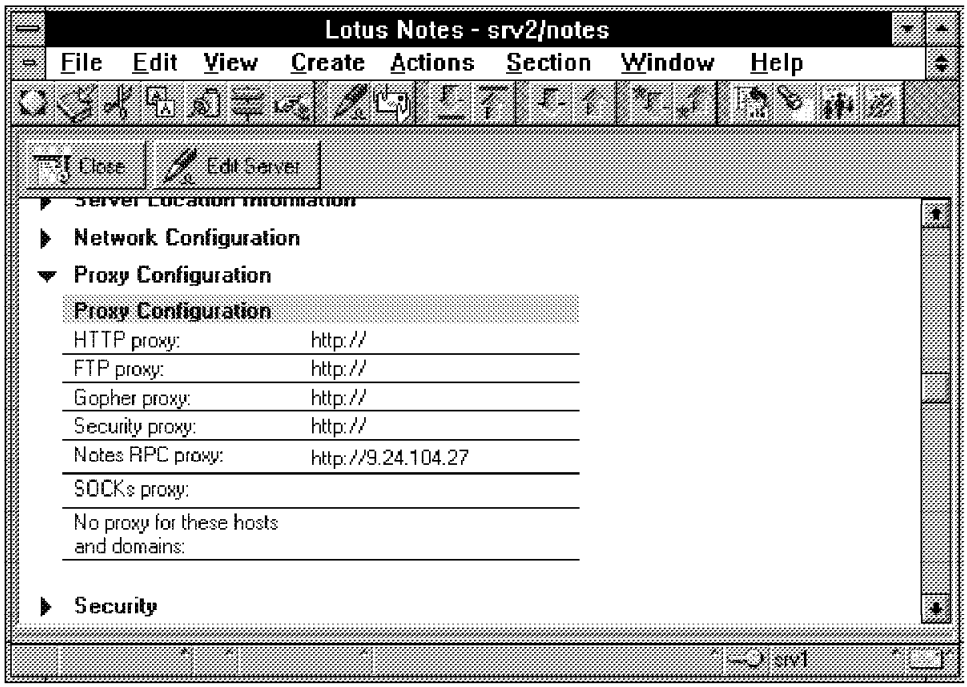


Figure 146. Defining the Proxy Server

2. The proxy server must support the CONNECT method. We used the Netscape proxy server for this purpose. On the proxy server, create a new filter for Lotus Notes connections. We modified the /usr/ns-home/proxy-rs60004-proxy/config/obj.conf file manually. The following is a fragment we inserted at the bottom of the obj.conf file:

   ```
   <Object ppath="connect://.*:1352">
   Service fn="connect" method="CONNECT"
   </Object>
   ```

   This entry tells the server to accept CONNECT requests for session with a target port of TCP/1352.

3. To verify the setting, open the administrator console (which is a Web browser interface on the Netscape proxy server) and select **System Settings** followed by **View Server Settings**.

Note that the screen allows you to decide how the session is to be routed to its destination. For example, you could choose to pass it through a SOCKS server.

7.5 Notes As a Secure Mail Gateway

With all the excitement generated by the World Wide Web, it is easy to forget that the most dominant application using the Internet is e-mail. The standard protocol for mail distribution is the Simple Mail Transfer Protocol (SMTP). If you use Notes mail internally in your organization, the Notes mail transfer agent (MTA) can allow you to give access to Internet mail in a secure and reliable way. Figure 147 on page 147 shows how the MTA works.

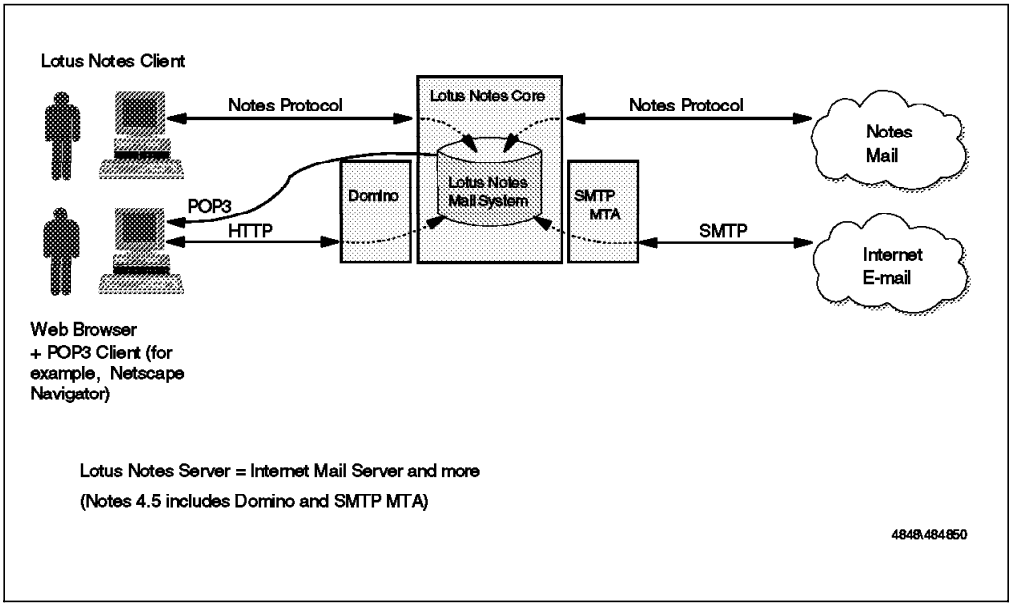


Figure 147. E-mail Connectivity in Notes 4.5

Notes mail uses the same server/server and server/client communications as any other Notes traffic. This means that you can place a Notes server with MTA outside the firewall boundary and then use any of the configurations described above to connect it to Notes mail in the internal network.

Appendix A. Cryptographic Techniques

Notes and Internet security mechanisms make use of a number of common cryptographic techniques. It is important to have a good understanding of these techniques and in the other chapters in this book we have assumed that you have some basic knowledge of them. However, this is a complex area, so in this section we present a brief overview of the important cryptographic techniques. We hope that you will find this a useful resource.

We discuss five subject areas:

1. Symmetric key (or bulk) encryption
2. Public key encryption
3. Secure hash (or digest) functions
4. Digital signatures and other combinations of the above techniques
5. Certification mechanisms

If you want to learn more about cryptography, we recommend the RSA Frequently Asked Questions document at http://www.rsa.com/rsalabs/newfaq.

A.1 Symmetric Key Encryption

This is a grown-up version of the kind of secret code that most of us played with at some time during childhood. Usually these use a simple character replacement algorithm; if you want to encrypt a message, you just replace each letter of the alphabet with another. For example:

```
Original letter: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Replacement    : GHIJKLMNOPQRSTUVWXYZABCDEF
```

In this case the letters in the alphabet have just been shifted seven places to the right, so HELLO WORLD would translate to NKRRU CUXRJ. The premise on which this code is based is that both the sender and the receiver know a common key, in this case the number of places to shift the letters. This *shared secret* allows the receiver of the message to reverse the encryption process and read the scrambled message. Symmetric encryption algorithms used by computers have the same elements as this simple example, namely a mechanism to scramble the message (also known as a cipher) and a shared secret (a key) that allows the receiver to unscramble the encrypted message.

The strength of a symmetric key cipher of this kind is dictated by a number of factors. For example, It is important that it effectively randomizes the input, so that two related clear-text messages do not produce similar encrypted results. Our childish example falls down badly in this area, because each letter always converts to the same encrypted result, and because it does not encrypt spaces. The kindergarten cryptanalyst can quite easily break the code by knowing that any one-letter word is likely to be an A. For full-strength symmetric ciphers, much of the work of the cryptanalyst involves trying to find patterns in the result of the algorithm, to use as a shortcut to breaking the code.

If the encryption algorithm has no flaws of this kind, the main factor governing its strength is the size of the *key space*; that is, the total number of possible shared secrets. Once again our simple example falls short, because it only has 25

© Copyright IBM Corp. 1997

possible keys. We could mount a *brute force* attack very easily by trying each key in turn until we found a message that makes sense. Real symmetric ciphers use numeric keys, usually of between 40 and 128 bits in size. Even for the smallest of these a brute force attack has to try, on average, 2 to the power 39 or about 550,000,000,000 possible keys. Each extra bit of key size doubles the key space.

A.1.1 Characteristics of Symmetric Key Algorithms

There are a number of symmetric key ciphers in use. We have listed the main ones below together with a brief description of their capabilities. However, they also share several common characteristics:

- They are fast and cause a relatively low system overhead. For this reason symmetric key encryption is often referred to as bulk encryption, because it is effective on large data volumes.
- The algorithms are published openly and there are no commercial licensing issues to be considered in implementing them.
- They all fall under the control of the U.S. National Security Agency export restrictions. The precise operation of these restrictions is not a simple matter, but in essence that means that:

  1. Any software incorporating cryptographic technology that is exported by a U.S. company has to have a special export license.
  2. If the product includes symmetric encryption code that can be used for encrypting an arbitrary data stream, the license will *only* allow unrestricted export *if* the key size is smaller than a given, NSA-specified, value.

  What this means is that to export full-strength cryptography, a company has to have a special license for each customer. Such licenses are only issued for customers that the U.S. government considers to be friendly, such as major banks and subsidiaries of U.S. companies. Until recently the threshold key size for a general export license was 40 bits. Several challenges have shown that a brute force attack can be mounted against a 40-bit key with relatively modest computing power. A government announcement in October 1996 opened the door to the use of larger keys, initially up to 56 bits, with the promise of unlimited key sizes when the computer industry develops effective key recovery technology. (Key recovery means that the key for a session can be discovered, given the knowledge of some other, master, key.) 56 bits may not sound a lot better than 40, but in fact it is 2 to the power 16, or 65,536 times more difficult to crack.

A.1.1.1 Common Symmetric Key Algorithms

DES The Data Encryption Standard is the most commonly-used bulk cipher. It was originally developed by IBM in 1977 and it has resisted all attempts at cryptanalysis. DES breaks the data to be encrypted into a sequence of 64-bit blocks and then uses a 56-bit key to apply a sequence of mathematical transforms to it. There are a number of variations on the standard DES algorithm, such as cipher block chaining, in which each block of data is XOR′d with the previous block before encryption, and triple-DES in which DES is applied three times in succession.

RC2/RC4 These related ciphers were developed by RSA Data Security, Inc. RC2 is a block cipher similar to DES, whereas RC4 operates on a stream of data. They both use a 128-bit key, but they support key masking. This means that part of the key may be set to a known value so that the effective key size is whatever remains from the 128-bit total. This has been used to good effect in producing 40-bit versions of software products for export.

IDEA The International Data Encryption Algorithm is another block cipher, in the mold of DES. It uses a 64-bit block size and a 128-bit key. IDEA is the bulk encryption algorithm used by Pretty Good Privacy (PGP).

A.2 Public Key Encryption

A non-mathematician can intuitively understand how a symmetric key cipher works by extrapolating from a familiar base. However, public key mechanisms are much less accessible to the lay person. In fact, it sometimes seems more like magic than technology. The fundamentals of public key encryption are:

1. Instead of a single encryption key, there are two related keys, a *key pair*.
2. Anything encrypted using one of the two keys can *only* be decrypted with the other key of the pair.

Let us say that two people, Bob and Alice, want to exchange messages using a public key algorithm. Alice generates a key pair and places one of the keys, the *private key* in a safe place. She sends the other half of the key pair (called, naturally, the public key) to Bob. Bob can now encrypt a message using the public key and only the owner of the matching private key, Alice, can decrypt it. Of course, if Alice wants to send a reply, Bob needs to create his own key pair and send the public key to Alice. The big advantage of this mechanism over the symmetric key mechanism is that there is no longer any secret to share. In fact, it does not matter who has the public key, because it is useless without the matching private key.

There is one further advantage that public key gives us. In the above example, imagine that Alice uses her private key to encrypt a message and sends it to Bob. The message that is sent between them is still scrambled, but it is no longer private, because anyone with the public key can decrypt it (and we have said that we do not care who has the public key). So, what *can* we use this message from Alice for? The answer is: authentication. Because only Alice has access to the private key that created the message, it can *only* have come from her.

A.2.1 Characteristics of Public Key Encryption

Public key algorithms can trace their ancestry back to the Diffie-Hellman key exchange mechanism. Diffie-Hellman is not a general purpose encryption scheme, but rather a method of exchanging a secret key. There is only one widely-used general purpose public key mechanism, the Rivest, Shamir and Adelman (RSA) algorithm, which is the property of RSA Data Security, Inc. Public key, like any encryption mechanism, relies on the fact that certain mathematical problems are very hard to solve. The problem that the RSA algorithm relies on is the factorization of large numbers. For a detailed description of the mathematics behind RSA public key encryption, refer to http://www.rsa.com/rsalabs/newfaq/q8.html.

Clearly, public key has a big practical advantage over symmetric key, in that there is no need to securely share a secret between the sender and receiver. However, the RSA algorithm is a much less efficient encryption technique than any commercial symmetric key algorithm, by a factor of 100 or more. It is therefore not a good choice for encryption of bulk data.

RSA is subject to the same US export restrictions as symmetric algorithms. However, the key in this case is actually a very large number. Very approximately, an RSA key size of 1024 bits corresponds to a full-strength symmetric key of 64 bits or more.

A.3 Secure Hash Functions

The third tool in our encryption armory is not actually an encryption mechanism at all. A secure hash is a way of creating a kind of “fingerprint” of a message. A secure hash function has three main attributes:

1. It takes a message of any size and generates a small, fixed size, block of data from it (called a *message digest*). Re-executing the hash function on the same source data will always yield the same resulting digest.
2. It is not predictable in operation. That is to say, a small change in the source message will have an unpredictably large effect on the final digest.
3. It is, for all intents and purposes, irreversible. In other words there is no way to derive the source data, given its digested form.

What use is a secure hash function? Its main function is to detect whether a piece of data has been modified or not. We see in A.4, “Combinations of Cryptographic Techniques” how this is used in combination with RSA to generate a digital signature.

There are two secure hash algorithms in common use. The most widely-implemented is MD5, which was developed by RSA Data Security, Inc. This generates a 128-bit digest from any length of input data. It is described in RFC1321. The other algorithm that is becoming increasingly common is the U.S. government-developed Secure Hash Standard (SHS). This produces a 160-bit digest, slightly larger than MD5.

A.4 Combinations of Cryptographic Techniques

In Chapter 4, “World Wide Web Security Features” on page 37 and Chapter 5, “Security in the Domino Server” on page 51 we describe the security protocols of Notes and the World Wide Web. Although they differ in detail they both use the three techniques above. Two combinations of the techniques are used very commonly, so we outline them here:

- Public key delivery of a symmetric key

  It is most efficient to use a symmetric key algorithm for bulk data, but first you have to copy the key from sender to receiver. A common approach is to use a public key algorithm to protect the key transfer process.

- Digital Signatures

  You do not always want to encrypt data in transit. Very often the contents of a message may not be secret, but you do want to be sure that it really came from the apparent sender. If Alice wants to prove to Bob that it was really

she who sent him a message she will attach a signature to the message, exactly as she would if writing a real letter on paper. In the digital case she creates the signature by first generating a digest of the message and then encrypting the digest with her private key. When Bob receives the message he first decrypts the digest (with Alice′s public key) and then generates his own digest from the received message. If the two digests match, Bob knows that:

1. The message is the same as when it was sent (because the digest is the same).
2. It really was sent by Alice, because only she has the private key.

For a more detailed example of how Notes uses digital signatures, see Figure 18 on page 29.

A.5 Public Key Certificates

We have seen how public key cryptography overcomes the problem of having to pass a secret from sender to receiver. There is still a need to send a key, but now it is a public key, that anyone can see because it is only useful to an attacker if he also has the private key. However, this overlooks one crucial element of trust: how can you be sure that the public key really came from who you think it came from?

One answer is to only pass public keys to someone you know. Bob and Alice have known each other for a long time (in fact people have started to talk), so they could share their public keys by exchanging diskettes. For normal cases, however, you need some way to be sure that a public key is authentic.

The mechanism for doing this is the *public key certificate*. This is a data structure containing a public key, plus details of the owner of the key, all digitally signed by some trusted third party. Now when Alice wants to send Bob her public key she actually sends a certificate. Bob receives the certificate and checks the signature. As long as it has been signed by a certifier that he trusts, he can accept that this really is Alice′s key.

Certificates in real life are more complex than this. Descriptions of how they are used in a variety of ways in the detailed sections about SSL and Notes security elsewhere in this book.

Appendix B. Special Notices

This publication is intended to help technical professionals, planners, network designers and consultants to understand the security implications of the Lotus Domino solution. The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Notes or the Domino server.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	Client Access
WebExplorer	IBM

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

© Copyright IBM Corp. 1997

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

DCE	The Open Software Foundation
Domino	Lotus Development Corporation
Lotus Notes	Lotus Development Corporation
Lotus	Lotus Development Corporation
LotusScript	Lotus Development Corporation
Netscape Navigator	Netscape Communications Corporation
Netscape	Netscape Communications Corporation
Notes	Lotus Development Corporation
OpenView	Hewlett-Packard Company
Replication	Lotus Development Corporation
Tivoli	Tivoli Systems Inc., an IBM Company
Tivoli Management Environment	Tivoli Systems Inc., an IBM Company
TME 10	Tivoli Systems Inc., an IBM Company
IPX	Novell, Incorporated
Windows NT	Microsoft Corporation
NotesView	Lotus Development Corporation
Intel	Intel Corporation

Other trademarks are trademarks of their respective companies.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 159.

- *www.security: How to Create a Secure World Wide Web Connection*, SG24-4564
- *Building a Firewall with the IBM Internet Connection Secure Network Gateway*, SG24-2577
- *Enterprise-Wide Security Architecture and Solutions*, SG24-4579

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

These publications are also relevant as further information sources:

- *Lotus Notes: Administrator's Guide*, Part No. 12755
- *Lotus Notes: Programmer's Guide Part 1*, Part No. 12752
- *Lotus Notes: Programmer's Guide Part 2*, Part No. 12753
- *Lotus Notes: Application Developer's Guide*, Part No. 12750
- *Lotus Notes: Working with Lotus Notes and the Internet*, Part No. 12763
- *Lotus Notes Network Design*, SR23-7378

© Copyright IBM Corp. 1997

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

© Copyright IBM Corp. 1997

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
(+45) 48 14 2207 (long distance charge)	Outside North America

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Index

Special Characters

Numerics

A

B

C

D

© Copyright IBM Corp. 1997

E

F

G

H

I

K

T

U

V

W

ITSO Redbook Evaluation

The Domino Defense: Security in Lotus Notes and the Internet
SG24-4848-01

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

© Copyright IBM Corp. 1997

Printed in U.S.A.